Title

End-to-End Joint Image Compression and Deep Learning under Bandwidth Constrained Environments

Author

Hewa Gamage, Lahiru D Chamain

Publication Date

2022

Peer reviewed|Thesis/dissertation

End-to-End Joint Image Compression and Deep Learning under Bandwidth Constrained Environments

By

LAHIRU DULANJANA CHAMAIN HEWA GAMAGE

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

_____

Zhi Ding, Chair

_____

Soheil Ghiasi

_____

Lifeng Lai

Committee in Charge

2022

Abstract

End-to-End Joint Image Compression and Deep Learning under Bandwidth Constrained Environments

The past decade has witnessed the rising dominance of deep learning (DL) and artificial intelligence (AI) in a wide range of applications. In particular, the ocean of wireless smart phones and IoT devices continue to fuel the tremendous growth of edge/cloud-based machine learning (ML) systems including image/video recognition and classification. To overcome the infrastructural barrier of limited network bandwidth in cloud ML, existing solutions have mainly relied on traditional compression codecs such as JPEG that were historically engineered for human-end users instead of ML algorithms. Traditional codecs do not necessarily preserve features important to ML algorithms under limited bandwidth, leading to potentially inferior performance. This dissertation investigates application-driven optimization of programmable commercial codec settings for networked learning tasks such as image classification.

In the first part of this dissertation, we focus on the efficient use and optimization of existing off-the-shelf commercial image compression codecs in bandwidth constrained image classification applications. We consider a cloud-based inference application where a power and memory limited embedded source device transmits the collected images to a powerful cloud server over bandlimited wireless channels. Our main contributions are two folds. Firstly, we show that the reconstruction step of the existing image decoders is unnecessary for cloud-based inference. Deep learning classifiers designed to take intermediate features as inputs, instead of RGB images, can perform inference few times faster with the same or improved classification accuracy. Secondly, we show that redesigning the entropy coders of commercial image codec such as JPEG2000 and learning optimal parameter setting of the entropy coders for a given task in end-to-end manner can significantly improve rate-accuracy performance of the codec.

In the second part, we investigate the methods of improving rate-distortion-accuracy performance in cloud-based AI applications for DL-based image compression codecs. Exploring

end-to-end optimization of the complete codec, we propose novel classifier architectures based on variational auto-encoders (VAE) that outperform rate-classification accuracy of several conventional codecs. Further investigating DL-based codecs, we discuss how to achieve better rate-distortion-accuracy performance with end-to-end training revisiting the concept of region of interest (ROI).

In the third part of this dissertation, we explore recent interpretable information theory based concepts when modeling real world data and their applicability in data constrained deep learning scenarios. In particular, we investigate the use of linear discriminative representations (LDR) of images when designing cloud-based deep learning systems with improved rate-accuracy performance. Further, considering challenging but practical data constrained tasks such as zero-shot and few-shot learning, we investigate the generalization of such linear feature representations learned with rate reduction concepts.

Publications

The following publications are included as a whole or in parts in this dissertation:

- Chamain, L.D., Qi, S. and Ding, Z., 2022. End-to-End Image Classification and Compression with variational autoencoders. IEEE Internet of Things Journal, 9(21), pp. 21916-21931.
- Chamain, L.D., Qi, S. and Ding, Z., 2021, August. An End-to-End Learning Architecture for Efficient Image Encoding and Deep Learning. In 2021 29th European Signal Processing Conference (EUSIPCO) (pp. 691-695). IEEE.
- Chamain, L.D. and Ding, Z., 2020, May. Improving deep learning classification of JPEG2000 images over bandlimited networks. In 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 4062-4066). IEEE.
- Chamain, L.D., Cheung, S.C.S. and Ding, Z., 2019, July. Quannet: Joint image compression and classification over channels with limited bandwidth. In 2019 IEEE International Conference on Multimedia and Expo (ICME) (pp. 338-343). IEEE.

Further, the following articles were written during my PhD research period, but not included in this dissertation:

- Chamain, L.D., Racapé, F., Bégaint, J., Pushparaja, A. and Feltman, S., 2021, March. End-to-end optimized image compression for machines, a study. In 2021 Data Compression Conference (DCC) (pp. 163-172). IEEE.
- Chamain, L.D., Racapé, F., Bégaint, J., Pushparaja, A. and Feltman, S., 2021. End-to-end optimized image compression for multiple machine tasks. arXiv preprint arXiv:2103.04178.

Acknowledgments

I would like to thank my advisor Prof. Zhi Ding for giving me the freedom to choose the research path that interests me and allowing me to follow that path for five years while being financially supported. His invaluable guidance made me a better student, writer and the researcher that I'm today. I'm truly grateful for his contribution in revising my manuscripts and providing technical insights. I really appreciate his support through the past pandemic which helped me immensely to continue my research work with sound body and mind.

I would like to thank the members of my qualifying examination committee for their encouraging comments, kind advices and insightful questions. Thank you Prof. Samson Cheung for the early discussions we had on our QuanNet paper. They helped me to continue the project with confidence. I am also indebted to the members of my dissertation committee, Prof. Soheil Ghiasi and Prof. Lifeng Lai, for the time they spent on reading my dissertation and their encouraging comments. I have taken classes from both of them and enjoyed the well spent hours.

I would also like to thank my research collaborator Siyu Qi for her help revising my manuscripts, debugging the codes and great many hours we spent on discussions throughout past years. I'm equally thankful to the members of BRAT lab at UC Davis for welcoming me and making the lab a friendly place. Thank you Carlos for the cheerful conversations we had over the years that helped me keep going. I'm grateful to all my friends in Davis for supporting and encouraging me, specially for making myself feel at home.

None of my accomplishments would have been possible without the everlasting love and support of my family. The caring and guidance of my beloved parents, Nalini and Piyarathna, and my sisters, strengthened me to achieve my life goals. Special thanks to Dr. Prathapasinghe Dhramawansa for encouraging me to pursue graduate studies in California.

The work invested into this dissertation was supported in part by the National Science Foundation under Grants No. 2002927 and No. 2002937.

CONTENTS

Abstract ii

Publications iv

Acknowledgments v

Abbreviations xviii

1 INTRODUCTION 1

1.1 Image compression 3

1.1.1 Optimization of conventional compression codecs 3

1.2 Learning-based Image Compression 4

1.2.1 Learning-based codecs for improved rate-distortion-accuracy 5

1.3 Practical limitation of cloud-based inference systems 6

1.3.1 LDRs for joint compression and classification 8

2 BACKGROUND 10

2.1 Conventional image compression codecs: JPEG 2000 10

2.1.1 JPEG2000 Encoder 10

2.1.2 JPEG2000 Decoder 14

2.2 Learning-based codecs: Variational Autoencoders (VAE) 15

2.2.1 Variational Bound for Reconstruction 17

2.3 Linear Discriminative representation (LDR) 20

2.3.1 Maximal Coding-Rate-Reduction (MCR^2) 21

I IMPROVING CONVENTIONAL CODECS

3 FASTER AND ACCURATE DEEP LEARNING FOR JPEG2000 IMAGES 24

3.1 Introduction 24

3.1.1 Related works: Image classification on DWT coefficients 25
3.1.2 Is reconstruction necessary? 26
3.2 Classification of JPEG2000 Compressed images 27
3.2.1 DWT-specific Image Augmentation 27
3.2.2 Computation Speed and Accuracy 29
3.3 Experiments and Results 30
3.3.1 Shallow models are faster! 34
3.3.2 Experiments over Bandwidth Constrained Channels 35
3.4 Conclusions 35
4 JOINT OPTIMIZATION OF CONVENTIONAL J2K COMPRESSION AND DEEP LEARNING 37
4.1 Deep learning over bandwidth constrained channels 37
4.1.1 Related works 40
4.2 QuanNet: Joint Quantization and deep learning 40
4.3 Experiment Results 44
4.3.1 Truncation with DB1 wavelets 45
4.3.2 Quantization with DB1 wavelets 46
4.3.3 Quantization with CDF 9/7 wavelets 49
4.4 Discussion 50
4.5 Conclusions 52
II LEARNING-BASED CODECS
5 END-TO-END IMAGE CLASSIFICATION AND COMPRESSION WITH VARIATIONAL AUTOENCODERS 54
5.1 Introduction 55
5.2 Related Works: VAE-Based Classifier 59
5.3 A New VAE Classification Framework 61
5.3.1 Variational Bound and Loss for Classification 62

5.3.2 Learning Model . . . 64
5.3.3 Rate Loss . . . 65
5.3.4 Classification Loss . . . 66
5.4 Experiments and Results . . . 67
5.4.1 VAE-based Joint Compression and Classification models . . . 68
5.4.2 CIFAR-10 Experiments . . . 70
5.4.3 CIFAR-100 Experiments . . . 71
5.4.4 ImageNet Experiments . . . 72
5.5 Complexity Comparison . . . 72
5.5.1 Model size comparison . . . 73
5.5.2 Inference speed comparison . . . 75
5.5.3 Power savings comparison . . . 76
5.5.4 Comparison to Torfason-2018 . . . 77
5.6 Discussion: Joint Compression and Classification with Reconstruction . . . 78
5.6.1 Visualization of reconstructed images . . . 81
5.7 Conclusions . . . 82
6 CLASSIFICATION-GUIDED ROI-BASED END-TO-END IMAGE COMPRESSION 83
6.1 Introduction . . . 84
6.2 Related Works . . . 87
6.2.1 ROI Coding . . . 87
6.2.2 ROI Prediction . . . 89
6.2.3 Joint image compression and classification . . . 90
6.3 Method . . . 91
6.3.1 ROI-based image encoding . . . 91
6.3.2 ROI prediction with Saliency maps . . . 94
6.3.3 Classification-guided ROI compression . . . 95
6.4 Experiments and Results . . . 96

6.4.1 Details on Training . . . 97
6.4.2 Results and Comparisons . . . 99
6.5 Complexity comparison . . . 106
6.5.1 Model complexity . . . 106
6.5.2 Inference speed . . . 108
6.5.3 FLOPS calculation . . . 108
6.6 Discussion: Learned masks from shared encoder backbone . . . 109
6.7 Conclusions . . . 111

III LDR-GUIDED CODECS

7 LINEAR DISCRIMINATIVE REPRESENTATION (LDR)-GUIDED LOW-SHOT LEARNING 113
7.1 Introduction . . . 114
7.2 Background and Related Works . . . 116
7.2.1 ALE for Zero-Shot Learning . . . 116
7.2.2 ProtoNets for Few-Shot Learning . . . 117
7.3 Methods . . . 118
7.3.1 LDR-Guided Zero-Shot Learning . . . 118
7.3.2 LDR-Guided Few-Shot learning . . . 122
7.4 Experiments and Results . . . 123
7.4.1 LDR-Guided Zero-Shot Learning . . . 123
7.4.2 LDR-Guided Few-Shot Learning . . . 126
7.4.3 $\mathcal{R}$-$\mathcal{C}$ Performance on JPEG-Compressed Images . . . 129
7.5 Conclusions . . . 130
8 LDR-BASED HIERARCHICAL GROUPING FOR JOINT IMAGE COMPRESSION AND CLASSIFICATION 131
8.1 Introduction . . . 131
8.2 Background and Related works . . . 132

8.2.1 Non-asymptotic rate-distortion with multiple groups 133
8.2.2 Maximal coding for rate reduction (MCR^2) 134
8.2.3 MCR^2 and Latent Encoding 134
8.2.4 Latent Compression in View of Grouping 135
8.3 Proposed End-to-End Framework 136
8.3.1 Rate under Quantization Noise 137
8.3.2 Learning to Group 139
8.4 Experiments and Results 140
8.4.1 DL Network Architectures 141
8.4.2 Quantization Noise Emulation 141
8.4.3 Ablation Experiment: Distortion Learning 142
8.4.4 Ablation Experiment: Learned Grouping 143
8.4.5 Gaussian noise 143
8.5 Conclusions 144
9 CONCLUSIONS AND FUTURE WORK 145
9.1 Summary of contributions 145
9.1.1 Open source codes 146
9.2 Limitations and future work 146
9.2.1 Re-training/fine-tuning is necessary for each rate-accuracy point . . . 146
9.2.2 Universal encoders 148
Appendix A DATA SETS 149
A.1 Classification data sets 149
A.1.1 ImageNet-1k 149
A.1.2 Imagenette 149
A.1.3 Tiny-ImageNet 150
A.1.4 ModelNet-40 150
A.1.5 CIFAR-10 150

A.1.6 CIFAR-100 . . . 150
A.1.7 AwA2 . . . 150
A.1.8 CUB . . . 151
A.1.9 SUN . . . 151
A.2 Compression data sets . . . 151
A.2.1 HKU-IS . . . 151
A.2.2 DAVIS-2016 . . . 151
A.2.3 CADDY . . . 152
A.2.4 ASL . . . 152
Appendix B FASTER AND ACCURATE DEEP LEARNING FOR JPEG2000 IMAGES 153
B.1 ResNet for classification . . . 153
B.2 Fine-tuning on pre-trained models . . . 155
B.3 3D model classification with multi-view CNNs . . . 155
B.4 ResNet architectures used for DB1, DWT coefficients. . . . 158
Appendix C VAE-BASED CLASSIFIERS 159
C.1 Robustness to visual corruptions . . . 159
C.2 Implementation details: Effect of β adjustment . . . 160
C.3 Visualization of latent maps . . . 160
C.4 Further theoretical explanations . . . 162
C.4.1 Relationship to the Information Bottleneck . . . 162
C.4.2 An alternative problem formulation . . . 165
Appendix D CLASSIFICATION-GUIDED ROI-BASED END-TO-END IMAGE COMPRESSION 168
D.1 PoolNet generated ROI . . . 168
D.2 Complexity comparison . . . 168
D.3 Cai-2019 complexity estimation . . . 169
D.4 Classifier fine-tuning . . . 169

Appendix E LDR-GUIDED LOW-SHOT LEARNING 172

E.1 Classes versus Attributes: An ablation study 172

E.2 Robustness to Noise and common Perturbations 173

E.3 Robustness to number of unseen classes 173

E.4 LDR-Guided FSL Training 175

E.5 FSL Encoder Models 176

LIST OF FIGURES

Figure 1.1 Cloud AI applications. 2
Figure 1.2 Overview of learning-based image compression codec 4
Figure 1.3 Edge computing for Deep learning. 7
Figure 2.1 JPEG2000 encoder. 10
Figure 2.2 Level-1 DWT of YCbCr channels 12
Figure 2.3 Level-2 DWT coefficients with DB1 wavelet 13
Figure 2.4 JPEG2000 decoder for lossy compression. 15
Figure 2.5 General structure of a VAE 16
Figure 2.6 Modeling complex nonlinear data with LDR 20
Figure 3.1 Conventional training and inference for cloud-based image classification 24
Figure 3.2 Extracting DWT coefficients from within the J2k decoder 26
Figure 3.3 Image augmentation in DWT domain 28
Figure 3.4 Test accuracy vs inference and training speed for CIFAR-10 31
Figure 3.5 Proposed modified ResNet for CDF 9/7 inputs 33
Figure 3.6 Shallow models are faster! 34
Figure 3.7 Rate-accuracy and inference speed comparison for CIFAR-10 35
Figure 4.1 JPEG2000 lossy encoding process. 37
Figure 4.2 QuanNet 41
Figure 4.3 Illustration of Quan block 42
Figure 4.4 A differentiable dead zone quantization approximation 42
Figure 4.5 Proposed cloud-based training and inference with QuanNet 44
Figure 4.6 Classification accuracy for DB1 DWT degrades with limited BW . . 47

Figure 4.7 Rate-accuracy performance of QuanNet with DB1 wavelet 48
Figure 4.8 Rate-accuracy performance of QuanNet with CDF 9/7 wavelet . . . 50
Figure 4.9 Visual quality of the reconstructed images 52
Figure 5.1 An illustration of a networked ML instance 55
Figure 5.2 Proposed VAE-based classification framework 57
Figure 5.3 Overview of the proposed VAE classifier during inference 62
Figure 5.4 Rate-accuracy results for CIFAR-10 and CIFAR-100 68
Figure 5.5 Encoders and classifiers designs the proposed VAE-based classifier . 69
Figure 5.6 Rate-accuracy performance on ImageNet-1k 71
Figure 5.7 Proposed joint classification and compression model with reconstruction 79
Figure 5.8 Proposed Decoder design for VAE-based classifiers 80
Figure 5.9 Examples of CIFAR-10 and ImageNet-1k reconstructed images . . . 81
Figure 6.1 Distortion artifacts depend on the loss function used during training 85
Figure 6.2 General architecture of existing ROI coding in learning based codecs 88
Figure 6.3 Proposed feature-sharing ROI prediction 90
Figure 6.4 Overview of the proposed classification-guided ROI image compression 92
Figure 6.5 Rate-distortion results for HKU-IS data set 99
Figure 6.6 $\mathcal{R}$-$\mathcal{D}$ results for different mask sizes of DAVIS-2016 100
Figure 6.7 $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ results for Imagenette data set 101
Figure 6.8 $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ results for ImageNet-1k data set 102
Figure 6.9 $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance for CADDY data set 103
Figure 6.10 Visualization of generated masks for CADDY data set 104
Figure 6.11 Learned masks for ASL data set 105
Figure 6.12 $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance for ASL data set 106
Figure 6.13 Classification-guided ROI masks 110
Figure 7.1 An overview: LDR-guided Zero-Shot Learning 115
Figure 7.2 LDR-guided FSL model 117

Figure 7.3 LDR-guided ZSL model 119
Figure 7.4 Examples of class vs attribute -based membership 120
Figure 7.5 Ablation study: ZSL and GZSL accuracy comparison on AwA2 . . . 124
Figure 7.6 Conventional cloud-based ZSL on JPEG compressed images 129
Figure 7.7 $\mathcal{R}$-$\mathcal{C}$ performance for AwA2 under the LDR-guided framework 129
Figure 8.1 Proposed end-to-end framework 136
Figure 8.2 Encoder and classifier architectures for CIFAR-100 and ImageNet-1k 137
Figure 8.3 $\mathcal{R}$-$\mathcal{C}$ on CIFAR-100 and ImageNet-1k under quantization noise . . . 140
Figure 8.4 Ablation results on CIFAR-10 142
Figure 8.5 Rate-accuracy for ImageNet-1k data set under Gaussian noise 144
Figure 9.1 Rate-accuracy comparison between single and multiple models . . . 147
Figure B.1 Building blocks for ResNet 153
Figure B.2 Fine-tuning on pre-trained models 155
Figure B.3 Multi View CNN model for 3D shape classification 156
Figure C.1 Robustness comparison for visual corruptions 159
Figure C.2 Rate-accuracy at different β values for CIFAR-10 161
Figure C.3 Visualization of latent maps 161
Figure D.1 ROI mask comparison for samples of HKU-IS data set 168
Figure D.2 $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ when fine tuning the classifier for Imagenette dataset 171
Figure E.1 ZSL and GZSL accuracy comparison. 172
Figure E.2 Noise severity vs accuracy performance of LDR-guided ALE 174
Figure E.3 Illustration of common noise and perturbations 174
Figure E.4 ZSL accuracy comparison of the proposed LDR-guided ALE 175

LIST OF TABLES

Table 3.1 Reconstruction time savings for different image sizes. 30
Table 3.2 Parameters of ResNet models for RGB and CDF 9/7 inputs 32
Table 3.3 Results of CIFAR-10 33
Table 3.4 Results of Tiny ImageNet. 33
Table 4.1 Effects of truncation (% of zeros) on classification. 45
Table 4.2 Effects of quantization on classification accuracy 46
Table 4.3 Quantization weights learned from QuanNet for ResNet-8 48
Table 4.4 QuanNet learned sub bands of CDF 9/7 coefficients. 51
Table 5.1 Summary of notations 61
Table 5.2 Complexity comparison of the models for CIFAR-10 and ImageNet . 73
Table 5.3 Inference speed comparison 75
Table 5.4 Number of FLOPs ($\times 10^9$) comparison for the proposed models. . . . 76
Table 5.5 Number of FLOPs comparison for the proposed models 78
Table 5.6 Reconstruction quality for CIFAR-10 and ImageNet-1k 81
Table 6.1 Complexity comparison in no. of parameters 107
Table 6.2 Inference speed comparison 108
Table 6.3 Computational power demand in terms of Flops in Billions (G). . . 109
Table 7.1 ZSL and GZSL accuracy comparison 125
Table 7.2 Ablation study: FSL accuracy comparison on mini-ImageNet data set 127
Table 7.3 LDR-guided FSL accuracy comparison with SOTA on mini-ImageNet 128
Table B.1 ResNet architecture for CIFAR-10 154
Table B.2 ResNet architecture for ImageNet 154

Table B.3 Results of ModelNet-40. 157
Table B.4 ResNet architecture for ModelNet-40. 157
Table B.5 ResNet architectures used for DB1, DWT coefficients. 158
Table D.1 Detailed complexity comparison in no. of parameters 169
Table D.2 Estimation of number of parameters of Cai-2019 170
Table E.1 LDR-guided FSL training details: learning rate and λ 175
Table E.2 LDR-guided FSL encoder architectures 176

ABBREVIATIONS

$\mathcal{R}$-$\mathcal{C}$ Rate-Accuracy.

$\mathcal{R}$-$\mathcal{D}$ Rate-Distortion.

$\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ Rate-Distortion-Accuracy.

AI Artificial Intelligence.

CE Cross Entropy.

CL Classifier.

CNN Convolutional Neural Networks.

D Decoder.

DL Deep Learning.

E Encoder.

EC Entropy Coder.

ED Entropy Decoder.

FSL Few Shot Learning.

LDR Linear Discriminative Representation.

MCR^2 Maximal Coding Rate Reduction.

ML Machine Learning.

MS-SSIM Multi-Scale Similarity Index.

MSE Mean Square Error.

PE Probability Estimator.

PSNR Peak Signal-to-Noise Ratio.

RGB Red, Green and Blue.

ROI Region of Interest.

VAE Variational AutoEncoder.

ZSL Zero Shot Learning.

1

INTRODUCTION

With the availability of massive amounts of labeled and unlabeled data, computational power with GPUs and parallel processing, and complex neural network (NN) models, the usage of deep learning (DL) has heavily increased in the areas including computer vision [2,3], wireless communication [4,5], medicine [6,7], economics [8,9] and security [10,11]. Deep learning has benefited substantially from the tremendous success of convolutional neural networks (CNNs) [12] in computer vision analysis applications such as image/video classification [13–16], object detection [17–19] and segmentation [20–22]. CNNs have proven to be effective in extracting low level features among adjacent pixels and formulating high level semantic features. The discovery of residual network (ResNet) architecture in [15] further solidified CNNs by overcoming the problem of classification/segmentation accuracy degradation with more layers exhibited in earlier versions of deep neural networks.

In many practical artificial intelligence (AI)-based mobile and wireless applications, the tasks of deep learning take place remotely from low cost embedded devices (source nodes) [23–25]. In the era of IoT, many typical multi-media analysis applications require the source node to transmit its acquired images/videos to a more powerful remote node as a part of cloud computing to perform the necessary DL inference tasks [26]. See Fig 1.1. With the growing use of Internet traffic for machine leaning purposes [27], networking industry has initiated several cloud machine learning platforms dedicated for deep learning[1].

Fig. 1.1 illustrates some of the common cloud-based learning and inference tasks: image classification, object detection, traffic monitoring, and road segmentation etc. In addition to inference tasks, visualization of the image/video data for human users is another major application of the cloud.

[1] https://www.networkworld.com/article/3305417/cisco-introduces-its-first-server-built-for-ai-and-ml-workloads.html

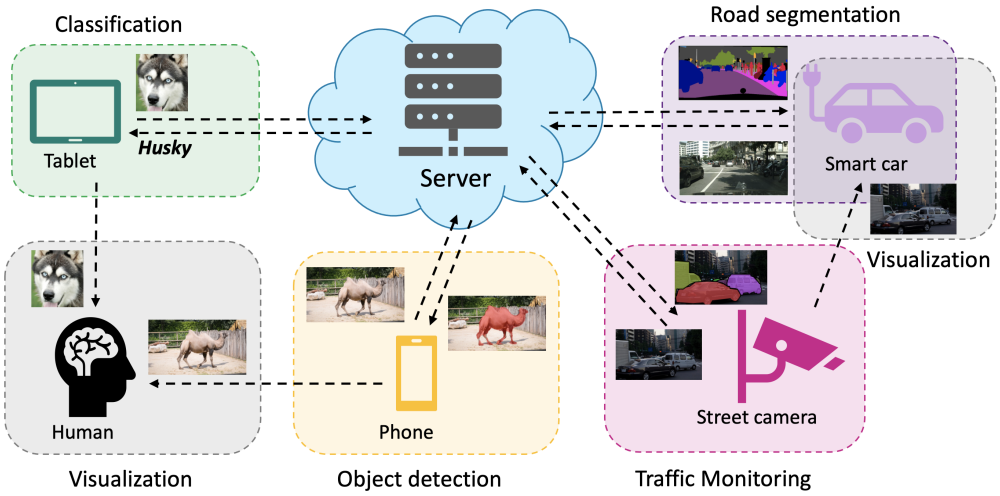


Figure 1.1: Cloud AI applications.

For each inference task, we identify two main processes: transmission of data from source to the server, and processing the received data at the server to perform the learning task. Due to the limited bandwidth (data rate) of the wireless channels, transmission of data, specially image and video that demand higher bandwidth in their raw format, needs to be compressed to a lower bandwidth formats before transmission [28]. This process is generally known as "image/video encoding". Once the data is compressed to a bit stream with manageable bandwidth, source device transmits it to either visualization devices for human perception or to a computational server for learning/inference purposes. Visualization devices "decode" the bit stream and reconstructs the image in visualizable formats such as RGB. The process of image encoding and decoding is collectively known as "image compression" and the combination of the encoding algorithm (encoder) and the decoding algorithm (decoder) is known as "codec". In this dissertation, we focus on the designing and joint optimization of the above two processes: image compression and DL task.

1.1 IMAGE COMPRESSION

Images captured by cameras contain Red, Green and Blue (RGB) pixel values corresponding to each point location of a 2D array. Transmitting or storing these raw pixel values can take a substantial amount of bandwidth/storage. For instance, an RGB image of size 256×256 in its raw pixel format assuming 8 bit color resolution takes at least 197 kB without any overhead. But the same image can be reproduced with 80kB including overhead by compressing the information contained in RGB values using image compression.

JPEG [28] is one of the most common lossy image compression standards that has been used for nearly 30 years. JPEG2000 [29], after being introduced 1998, has gained more popularity over the years due to its some of many fine qualities such as higher compression ratios, ability to decompress a once compressed image in many ways and region of interest compression. With this gaining popularity and usage, exploring ways to utilize these fine features for deep learning tasks is of timely importance. In Part i of this dissertation, we focus on how to utilize JPEG2000 compressed images and to optimize the compression parameters for deep learning tasks.

1.1.1 *Optimization of conventional compression codecs*

As illustrated in Fig. 1.1, for cloud-based DL applications like image/video classification, segmentation and object detection, a source device has to send the images/videos to the cloud servers to get the inference results. Common cloud AI systems compress these image/video data using the codecs like JPEG2000 or H264 before sending over the network. These image and video codecs however are designed and optimized to produce better visual quality outputs for human eyes at a given bandwidth, which is commonly known as "rate-distortion" ($\mathcal{R}$-$\mathcal{D}$) trade-off. We show that these conventional compression parameters are not optimal for deep learning tasks [30, 31] in Chapter 3. We then introduce an end-to-end trainable joint image

compression and deep learning network called 'QuanNet' [25] in Chapter 4, that can save significant amount of bandwidth by learning quantization step sizes for JPEG2000 codec. Our proposed concept of learning optimal parameter settings for engineered blocks in conventional codecs can be readily applied for more recent codecs such as BPG[2] and VVC [32] as well.

1.2 LEARNING-BASED IMAGE COMPRESSION

Due to the design and optimization flexibility, learning-based codec designs [33, 34] have gained recent popularity. In general, a leaning-based codec consists of an Encoder Analyzer (E), an Entropy Coder (EC), an Entropy Decoder (ED) and a Decoder Synthesizer (D) [31]. Fig 1.2 shows the general overview of such learning-based codec.

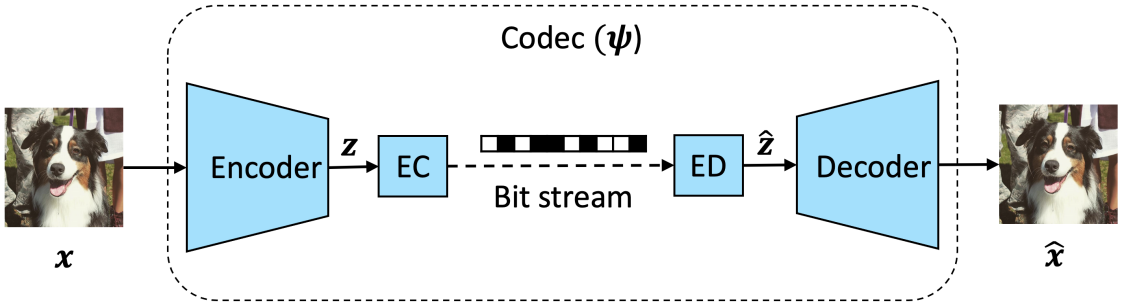


Figure 1.2: Overview of learning-based image compression codec. Input images (x) are transformed into latent $\boldsymbol{z}$, then quantized (Q) and entropy coded (EC). The binary stream is then stored or transmitted over a channel. On the receiver side, the decoder reconstructs the input images as $\hat{\boldsymbol{x}}$, from the decoded latent tensors $\hat{\boldsymbol{z}}$

.

In contrast to conventional codecs such as JPEG2000, each block of learning based codecs are designed with DL-based elements which can be optimized to minimize a given loss function. More importantly, all the blocks between the input image $\boldsymbol{x}$ and the reconstructed image $\hat{\boldsymbol{x}}$ of the codec as shown in Fig. 1.2 can be optimized in an end-to-end manner for a given objective such as rate-distortion. Such optimization of the complete codec towards a

[2] https://bellard.org/bpg/

common objective can naturally lead to higher rate-distortion performance as will be shown in following chapters.

1.2.1 *Learning-based codecs for improved rate-distortion-accuracy*

To overcome the infrastructural barrier of limited network bandwidth in cloud ML however, the existing solutions [33, 34], including the optimization of JPEG2000 discussed in Chapter 3-4, have mainly relied on traditional compression that were historically engineered for human-end users instead of ML algorithms. Traditional codecs do not necessarily preserve features important to ML algorithms under limited bandwidth, leading to potentially inferior performance [31].

Exploring the concept of learning-based compression, Part ii of this dissertation investigates application-driven optimization of programmable commercial codec settings for networked learning tasks such as image classification. Based on the foundation of variational autoencoders (VAEs) [35], in Chapter 5, we develop an end-to-end networked learning framework by jointly optimizing the codec and classifier for improved "rate-accuracy" ($\mathcal{R}$-$\mathcal{C}$) performance without reconstructing images for given data rate (bandwidth). We further show that a simple decoder can reconstruct images with sufficient quality without compromising classification accuracy.

As an extension of joint image compression and deep learning based on end-to-end optimization, in Chapter 6, we focus our investigation on designing learning-based codecs that achieve improved "rate-distortion-accuracy" ($\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$) performance simultaneously. For this purpose, we revisit the concept of region of interest (ROI) and its application in deep learning-based codecs.

ROI-based image compression achieves better $\mathcal{R}$-$\mathcal{D}$ performance for a given ROI region by allocating more bits compared to the rest of the image. For such ROI-based image compression applications without given ROI masks a priori, recent codecs based on deep learning (DL) have proposed effective step of ROI prediction, followed by subsequent ROI

encoding [36–38]. For network-based learning over cloud, however, limited bandwidth between sensor nodes and processing servers motivates the optimization of not only $\mathcal{R}$-$\mathcal{D}$ involving ROI prediction and image encoding, but also target DL task such as classification. Existing such end-to-end ROI prediction and coding frameworks still demand heavy computation, power and memory specially during feature extraction for ROI prediction [36, 38]. This causes lower inference speeds and shorter battery life of sensor nodes. Targeting practical applications, in Chapter 6, we propose significantly lighter-weight feature extraction backbones and to guide the feature extraction process with classification cross entropy loss. By doing so, we synergies ROI compression and classification tasks demonstrating improved $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance.

1.3 PRACTICAL LIMITATION OF CLOUD-BASED INFERENCE SYSTEMS

Given the explosive growth and deployment of IoT devices [39] and massive data collected by such devices [40], traditional cloud computing struggles to keep up with the demands of large IoT networks. Typical IoT configuration features source devices only as data collectors while cloud nodes are responsible for processing and analysis, hence, is limited by network link capacity, delay and losses, leading to long latency, unreliable inference and scalability issues [41–43]. Edge computing addresses these issues by deploying computing services in proximity to IoT devices [44]. Many image and video sensing devices such as smart phones, vehicular sensors and home/street cameras have sufficient resources to collect data and perform preliminary pre-processing and feature extraction, but not enough to implement full machine learning algorithms on-device [42]. With this new configuration of networked AI coupled with edge computing, such source devices can share a part of the computational burden by performing pre-processing and feature extraction, and deliver the extracted features to more powerful and resource rich edge/fog nodes for complex machine learning tasks.

Due to the layered architecture of the neural networks, we note three main properties of the data, passed from one layer to another, that can support the use of edge computing. See

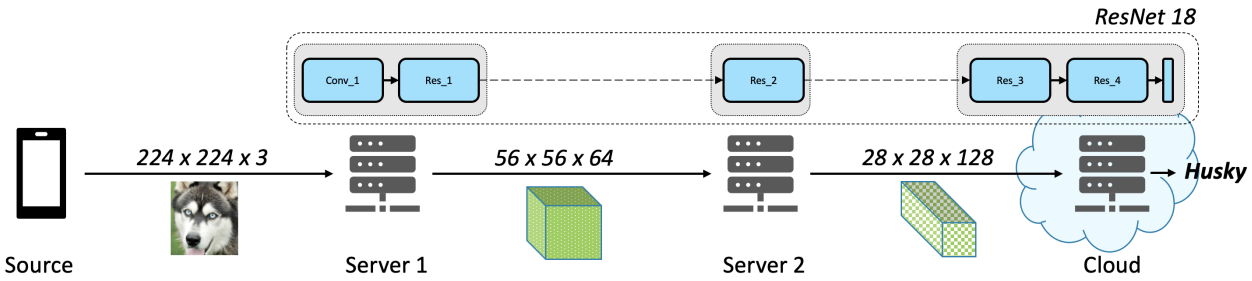


Figure 1.3: Edge computing for Deep learning.

Fig 1.3. 1). The redundancy of data is mostly reduced with layers by number of kernels, down sampling with strides and pooling. This reduces the amount of data to be sent to the central cloud for further computations. 2). The semantics between the output and the input to a layer are mostly different. Due to this, even if a data transmission is intercepted, the chance of interpreting the data can be low, preserving privacy of inputs [23]. 3) Low level features are common for task groups like classification, segmentation and object detection [15]. So the intermediate features computed by edge servers can be easily reused for multiple machine learning tasks without any modification to the existing deep models and communication networks.

Although edge computing has proven to be a powerful framework for tasks such as source device offloading [45] and content delivery [46], its use for analysis tasks has remained limited [27,47] due to non-generality of the features extracted or processed by the source devices and intermediate servers. Training intermediate data representations (latent embeddings) without careful supervision can lose essential features of input data resulting non-interpretable and non-universal features that can only be used for a specific task. Such features are less robust to common networked-AI related challenges such as missing or corrupted labels [48], data imbalance [49] and low-shot learning [50, 51]. Addressing such challenges, Part iii of this dissertation explores efficient latent embeddings for modeling real world image data.

1.3.1 *Linear discriminative representations for joint compression and classification*

Learning-based image/video codecs typically utilize the well known auto-encoder structure where the encoder transforms input data to a low-dimensional latent representation. Efficient latent embeddings can reduce bandwidth needs during compression for transmission and storage. However, finding such efficient latent embedding is quiet challenging considering the complexity of real-world-inputs and multiple post applications that rely on the learned latent embedding [52].

One natural approach to address this real-world-data transformation problem is to find learned representations with standard models or distributions. Along this direction, Chapter 7 explores one such family of learned distributions called "linear discriminative representation" (LDR) [53] where latent embeddings are distributed on linear, orthogonal group-wise subspaces. In particular, based on LDR, Chapter 7 proposes a novel classification framework when training with extreme data imbalance scenarios named "low-shot training". Focusing cloud-based networked classification applications, then we evaluate the proposed framework for rate-accuracy performance.

In zero-shot learning (ZSL) [50, 54, 55] and few-shot learning (FSL) [56–58], where training and inference are on disjoint classes, conventional cross-entropy (CE) based approaches tend to exhibit model over-fitting to seen classes. Hence, directly relying on seen class labels or attributes during training is disadvantageous. Recently proposed maximal coding-rate-reduction (MCR^2) principle [53] promotes LDRs with in-class compressive and inter-class discriminative properties by utilizing class labels as side information. When guided with LDR, encoders generate more universal features that are common to both seen and unseen classes. Evaluating over several benchmark data sets we show that such LDR-guided models achieve significant accuracy improvement over conventional ZSL and FSL baselines with zero added memory or computational complexity during inference.

Further investigating the $\mathcal{R}$-$\mathcal{C}$ performance of LDR-guided embeddings, Chapter 8 examines the effect of assigning high level coarse grouping labels to each learned latent subspace. We show that such grouping can be learned via end-to-end optimization of the codec and the DL model to optimize rate-accuracy for a given data set [59]. For cloud-based inference, source encoder can select a coding profile based on its learned grouping and encode the data features accordingly. Our test results on image classification show significant performance improvements with learned grouping over its non-grouping counterpart.

2

BACKGROUND

In this chapter, we provide background on topics such as JPEG2000 codec, variational autoencoders and linear discriminative representations which are foundational concepts used in following chapters of this dissertation.

2.1 CONVENTIONAL IMAGE COMPRESSION CODECS: JPEG 2000

JPEG2000 image compression codec mainly consists of an encoder and a decoder. Encoder takes RGB image as the input and produces a bit stream (j2k stream) for transmission or storage. Decoder takes the j2k stream as the input and reconstructs the RGB image. JPEG2000 offers a lossless and a lossy compression options corresponding to reversible and irreversible paths in Fig. 2.1.

2.1.1 *JPEG2000 Encoder*

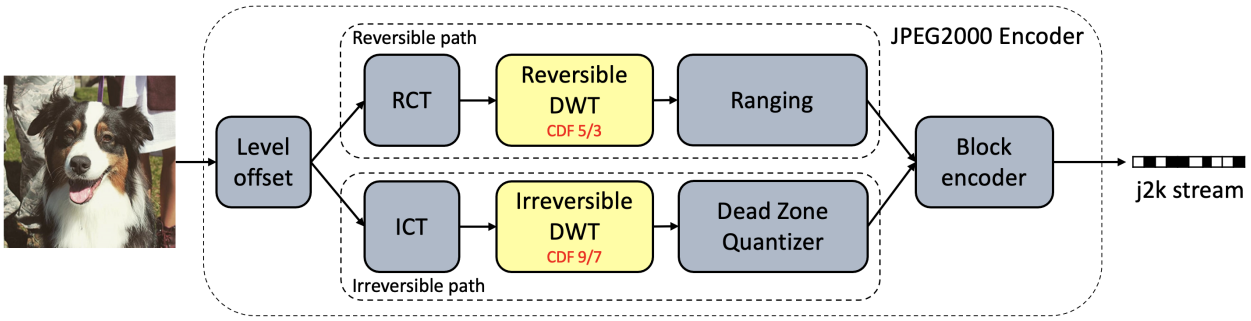


Figure 2.1: JPEG2000 encoder.

In this dissertation, we focus on the irreversible compression path and following are the main operations of the encoder.

Level offset

If the image sample values (unsigned) are B bits encoded, an offset of -2^{B-1} is added to center the pixel values. Ex: RGB pixels [0,255], are given a -128 offset.

Reversible/Irreversible Color transformation (RCT/ICT)

Color domain is changed to facilitate compression. In the reversible path, RGB values are transformed to YDbDr domain and in the irreversible path RGB values are transformed to YCbCr domain [29]. RGB pixel (x_R, x_G, x_B) can be converted to YCbCR value (x_Y, x_{Cb}, x_{Cr}), with the following linear transformation.

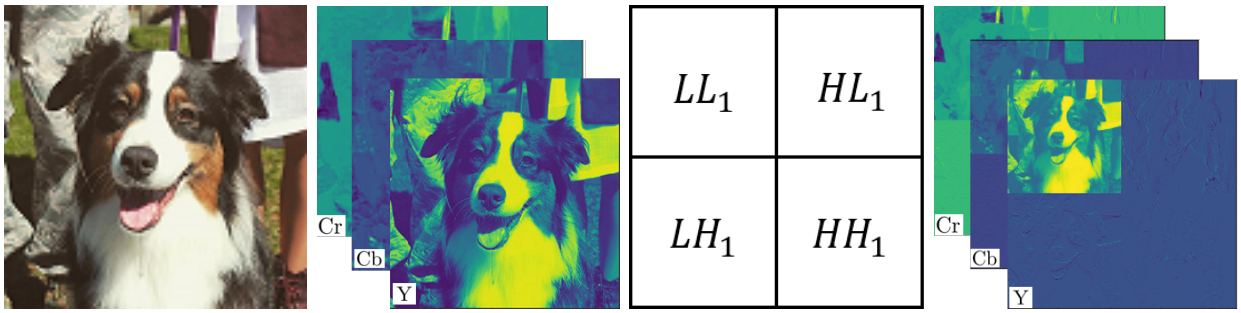
$$\begin{bmatrix} x_Y \\ x_{Cb} \\ x_{Cr} \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.3331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \times \begin{bmatrix} x_R \\ x_G \\ x_B \end{bmatrix} \tag{2.1}$$

x_Y which is a weighted average of R,G and B components, measures intensity (luminance). x_{Cb} and x_{Cr} which are the weighted differences between luminance and blue and, luminance and red respectively, measure chrominance. Since human eyes are known to be less sensitive to distortions in chrominance, these channels can be more compressed with less added distortion.

Discrete wavelet transformation (DWT)

Spatial coefficients (YCbCr/YDbDr) are transformed to spectral domain using a DWT. DWT can be described as a sequence of samples of the projection/convolution of a given signal $x(t)$ with a wavelet function $h(t)$. By appropriately choosing $h(t)$, different decompositions of the signal $x(t)$ can be obtained. For example in discrete domain, by choosing,

$$g[n] = \frac{1}{\sqrt{2}}[1, 1] \text{ and } h[n] = \frac{1}{\sqrt{2}}[1, -1],$$

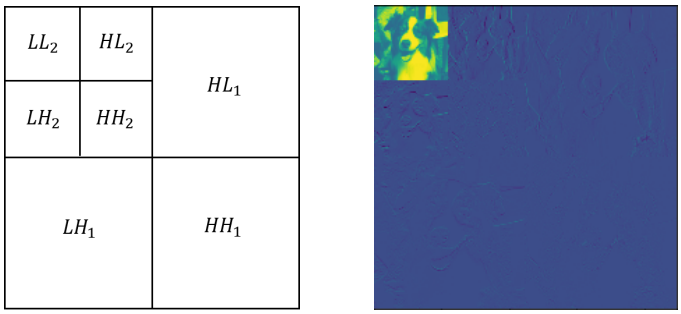


(a) (b) (c) (d)

Figure 2.2: (a) RGB image. 'Brownie' of size 64×64. (b) YCbCr channels of (a). (c) Sub-band structure of level-1 DWT transformed coefficients. Here LL_1 is the level-1 approximation of the original image and HH_1 represents highest frequency components. (d) shows level 1, DWT coefficients with DB1 wavelet of (b).

one can obtain sub sampled, bi-orthogonal low pass (L_1) and high pass (H_1) decompositions of signal $x[n]$ respectively. The above filters correspond to well-known 'Harr' or 'DB1' wavelet decomposition. In the context of 2-Dimensional (2-D) images, these filters can be applied in both horizontal and vertical directions resulting LL_1, LH_1, HL_1 and HH_1 decompositions (aka sub-bands). Here LL_1 is the level-1 approximation of the original image and HH_1 represents highest frequency components. For DB1 wavelets, the sub-bands LH_1, HL_1 and HH_1 represent horizontal, vertical and diagonal edges of a 2-D image respectively. See (b) and (d) of Fig. 2.2. Unlike Fourier transforms, DWT provides both frequency and spatial information of an image. Level-2 decomposition of a 2-D image can be obtained by taking the level-1 decomposition of the LL_1 sub-band. This creates LL_2, LH_2, HL_2 and HH_2. See Fig. 2.3.

Instead of DB1, JPEG2000 codec uses CDF 9/7 wavelets during lossy compression, which is a more complex waveform compared to DB1. Since DWT is a projection of an spatial domain input X_s, we can rewrite the 2-D DWT operation as follows.



(a) (b)

Figure 2.3: (a) Sub-band structure of level-2 DWT transformed coefficients. LL_2 is the level-2 approximation of the original image and (b) shows level 2, DWT coefficients with DB1 wavelet of Fig 2.2(b).

Let n be the image dimension. Define $X_s \in \mathcal{R}^{n\times n}$ and $X_{w_A} \in \mathcal{R}^{n\times n}$ where X_{w_A} is the DWT of X_s and A is the DWT matrix of a wavelet a. We can write X_{w_A} as separable column and row transformations (See [60] for more details).

$$X_{w_A} = A^T X_s A. \tag{2.2}$$

For CDF 9/7 wavelets, we can formulate A by using its "fast lifting" decomposition [61]. The lifting implementation of the 1-Dimensional DWT of CDF 9/7 wavelet consists of two predictions (P_1, P_2), each followed by an update function (U_1 and U_2). Then, the resulting matrix is de-interleaved (S) to form high and low frequency components. This implementations consists of five "for" loops for each operation.

We note that the above DWT matrix operation can be implemented as a matrix multiplication of three sub-functions: predictions, updates and de-interleaving. It can be denoted as

$$A = P_1 U_1 P_2 U_2 S. \tag{2.3}$$

Matrix A and its inverse calculated as given in Eq. (2.3)[1].

[1] python code for this transformations is available at https://github.com/chamain/Faster-and-accurate-classification-for-J2K

Dead zone quantization

The process of quantization, maps a range of values with floating point precision to an integer. JPEG2000 uses the following definition of the uniform deadzone quantizer [29]

$$q(x) = \text{sgn}(x)\left\lfloor \frac{|x|}{d_s} \right\rfloor.$$

Here, $q(x)$ is the quantization index for sub-band coefficient x whereas d_s is quantization step for sub-band s.

Block encoder

Block encoder encodes the DWT coefficients of the input image. The encoding process in JPEG2000 is performed in a block wise manner where the quantized DWT coefficients of an input image is divided into several blocks and encoded independently. Typical block sizes are 64×64 and 32×32. Encoding and ordering the coefficients is done using Embedded Block Coding with Optimal Truncation (EBCOT) algorithm [62] introduced by David Taubman in 2000. This algorithm allows the truncation of the encoded bit stream for a given block such that images with different quality can be obtained depending on the truncated bit length when decoding. These truncation lengths correspond to different points of the rate distortion characteristics of JPEG2000 encoder.

2.1.2 *JPEG2000 Decoder*

JPEG2000 decoder consists of a block decoder, a dequantizer, inverse DWT transformation (IDWT), YCbCr to RGB color mapping followed by a level offset of +128. See Fig. 2.4. Dequantizer re-scales the decoded DWT coefficients by their corresponding step sizes as explained in section 2.1. IDWT block transforms the dequantized DWT coefficients to spacial domain. Decoder then reconstructs the image after mapping YCbCr coefficients to RGB

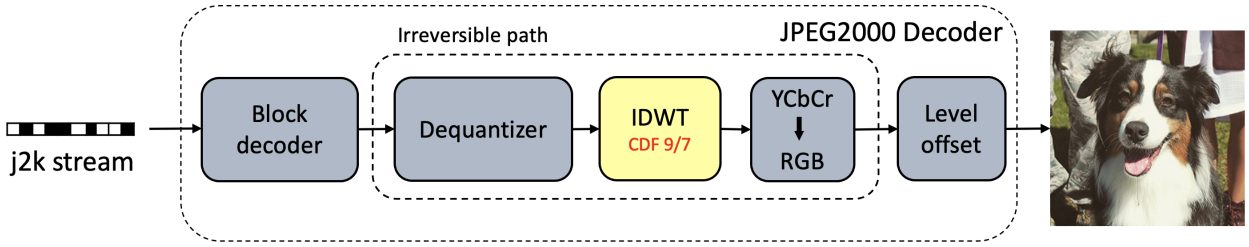


Figure 2.4: JPEG2000 decoder for lossy compression.

channels. In Chapter 3, we discuss how to skip some of these blocks when decoding leading to faster deep learning applications for j2k compressed images.

2.2 LEARNING-BASED CODECS: VARIATIONAL AUTOENCODERS (VAE)

When designing codecs for source compression to achieve optimized $\mathcal{R}$-$\mathcal{D}$ performance, accurate modeling of the data distribution is of vital importance. Depending on the underlying input distribution, the process of "generative modeling" can be challenging. Often, the input data ($\boldsymbol{X}$) manifold is high-dimensional and is complex to characterize.

Several likelihood-based methods have been proposed in the literature for generative modeling, such as auto-regressive models [63], flow-based methods [64, 65] and VAEs [35]. In this dissertation, we leverage the concept of VAE for joint image classification and compression. Without relying on strong assumptions, VAE exhibits fast training with back-propagation [66]. Such property is advantageous in comparison with model-based approaches relying on strong assumptions or requiring high computation complexity such as the Markov Chain Monte Carlo (MCMC) [35]. Further, VAE-based models are amenable to naturally interpretable loss terms that are directly related to rate-distortion trade-off in lossy compression [67], as shown later.

We can capture the general concept of VAE via Fig. 2.5. A VAE consists of an encoder for mapping the high dimensional input $\mathbf{x}$ into a latent representation $\mathbf{z}$, followed by a decoder in

charge of reconstructing the input that is denoted as input estimate $\hat{\mathbf{x}}$. The encoder's output $\mathbf{z}$ is a low-dimensional latent vector representing distinct features from the input data $\mathbf{x}$. The encoder functionality is to compress and to extract critical features by mapping input $\mathbf{x}$ into $\mathbf{z}$. The decoder can rely on $\mathbf{z}$ to reconstruct $\mathbf{x}$ for various remote applications. Thus, the VAE encoder makes it possible to store and transmit $\mathbf{z}$ instead of $\mathbf{x}$ to preserve bandwidth and storage.

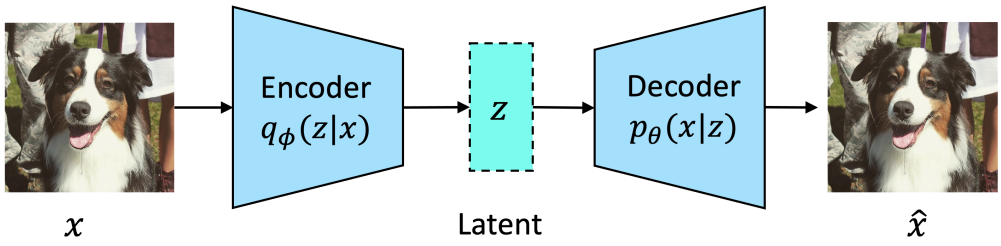


Figure 2.5: General structure of a VAE. Encoder transforms the input image $\boldsymbol{x}$ to a low dimensional latent vector $\boldsymbol{z}$. Decoder reconstructs image $\hat{\boldsymbol{x}}$ based on $\boldsymbol{z}$.

In the VAE formulation, the observable input data set $\mathcal{X}$ is assumed to consist of i.i.d. samples of $\mathbf{x}$ which are generated by some random process that involves an unobserved random variable $\mathbf{z}$ and generative model parameters $\boldsymbol{\theta}$ via

$$p_{\mathbf{x},\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z}) p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z}. \tag{2.4}$$

Neither the true parameters $\bar{\boldsymbol{\theta}}$ nor $\mathbf{z}$ are known.

Within this framework [35], the encoder can be viewed as to provide an efficient posterior estimate of the latent vector $\mathbf{z}$ from an observed input $\mathbf{x}$ for a given parameter setting $\boldsymbol{\theta}$. The decoder provides an approximate marginal inference of $\mathbf{x}$ upon reception of the latent vector $\mathbf{z}$ from the encoding transmitter.

2.2.1 *Variational Bound for Reconstruction*

VAE considers the general case when the posterior $p_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ is intractable for which an approximation $q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ parameterized by $\boldsymbol{\phi}$ is introduced to act as the encoder. For an arbitrary distribution $q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, we can derive the "variational bound" that follows the outlines of [35, 66] with detailed proof below for image reconstruction. Throughout the presentation, we denote an instance of the random variables $\mathbf{x}$ as $\boldsymbol{x}$. Assume $\boldsymbol{x} \in \mathcal{X}$ is a random sample of the random variable $\mathbf{x}$ which follows a generative model parameterized by $\boldsymbol{\theta}$ from an unobserved random variable $\mathbf{z}$. We can write

$$p_{\mathbf{x},\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z}) p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z}. \tag{2.5}$$

For an arbitrary distribution $q_{\mathbf{z},\boldsymbol{\phi}}(\boldsymbol{z})$, we can write,

$$p_{\mathbf{x},\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z}) p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z} = \int p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z}) \frac{q_{\mathbf{z},\boldsymbol{\phi}}(\boldsymbol{z})}{q_{\mathbf{z},\boldsymbol{\phi}}(\boldsymbol{z})} d\boldsymbol{z} = E_{q_{\mathbf{z},\boldsymbol{\phi}}} \left[p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \frac{p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z})}{q_{\mathbf{z},\boldsymbol{\phi}}(\boldsymbol{z})} \right].$$

Taking $-\log$ of both sides leads to

$$-\log p_{\mathbf{x},\boldsymbol{\theta}}(\boldsymbol{x}) = -\log E_{q_{\mathbf{z},\boldsymbol{\phi}}} \left[p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \frac{p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z})}{q_{\mathbf{z},\boldsymbol{\phi}}(\boldsymbol{z})} \right]. \tag{2.6}$$

Applying Jensen's inequality to (2.6), we write,

$$-\log p_{\mathbf{x},\boldsymbol{\theta}}(\boldsymbol{x}) \leq E_{q_{\mathbf{z},\boldsymbol{\phi}}} - \log \left[p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \frac{p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z})}{q_{\mathbf{z},\boldsymbol{\phi}}(\boldsymbol{z})} \right] \leq -E_{q_{\mathbf{z},\boldsymbol{\phi}}} \log p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - E_{q_{\mathbf{z},\boldsymbol{\phi}}} \log \left[\frac{p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z})}{q_{\mathbf{z},\boldsymbol{\phi}}(\boldsymbol{z})} \right].$$

Following the definition of KL divergence, we write,

$$-\log p_{\mathbf{x},\boldsymbol{\theta}}(\boldsymbol{x}) \leq -E_{q_{\mathbf{z},\boldsymbol{\phi}}} \log p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \mathrm{KL}(q_{\mathbf{z},\boldsymbol{\phi}}|p_{\mathbf{z},\boldsymbol{\theta}}) \tag{2.7}$$

Since $q_{\mathbf{z},\boldsymbol{\phi}}(\boldsymbol{z})$ is arbitrary, we can replace $q_{\mathbf{z},\boldsymbol{\phi}}(\boldsymbol{z})$ with conditional density $q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and write [35, 66],

$$-\log p_{\mathbf{x},\boldsymbol{\theta}}(\boldsymbol{x}) \leq -E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\left[\log p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] + \mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z},\boldsymbol{\theta}}) \tag{2.8}$$

where KL denotes the Kullback Leibler divergence.

This variational bound can serve as an optimization surrogate when direct minimization of $-\log p_{\mathbf{x},\boldsymbol{\theta}}(\boldsymbol{x})$ is intractable. Functionally, $q_{\mathbf{z}|\boldsymbol{x},\boldsymbol{\phi}}$ models the probabilistic encoder and $p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}$ models the reconstruction decoder as in Fig. 5.2(a). The first RHS term in Eq. (2.8) is the conditional entropy of $\mathbf{x}$ given $\mathbf{z}$ which quantifies the reconstruction loss.

The second RHS term $\mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z},\boldsymbol{\theta}})$ is related to the coding cost of the latents as shown later. Hence, the variational bound can be utilized to form the classical variational loss for reconstruction [35].

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\left[-\log p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] + \mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z},\boldsymbol{\theta}}) \tag{2.9}$$

When minimizing the variational bound as the loss function given in Eq. (2.9) that consists of two parts, a hard constraint can be imposed on the coding cost $\mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z},\boldsymbol{\theta}})$ while minimizing the reconstruction loss. To this extent, the authors of [68] further suggested adding a trade-off parameter $\beta(\geq 1)$ to Eq. (2.9) to reformulate a β-VAE loss function also adopted in [34, 67] as

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\left[-\log p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] + \beta\mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z},\boldsymbol{\theta}}). \tag{2.10}$$

Note that we can leverage the definition of cross entropy (CE) to rewrite

$$\mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z},\boldsymbol{\theta}}) = \mathrm{CE}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}, p_{\mathbf{z},\boldsymbol{\theta}}) - H(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}).$$

The term $\mathrm{CE}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}, p_{\mathbf{z},\boldsymbol{\theta}})$ averages the entropy $-\log(p_{\mathbf{z},\boldsymbol{\theta}})$ over the encoder distribution $q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}$ and captures the average encoding "cost" of latent representation $\mathbf{z}$. Following the approach of [67], if we consider only the deterministic encoder $\mathbf{z} = \boldsymbol{\phi}(\mathbf{x})$ for which $q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \delta(\boldsymbol{z} - \boldsymbol{\phi}(\boldsymbol{x}))$ and $H(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}) = 0$, we have

$$\mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z},\boldsymbol{\theta}}) = \mathrm{CE}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}, p_{\mathbf{z},\boldsymbol{\theta}}) = \mathrm{CE}(\delta(\boldsymbol{z} - \boldsymbol{\phi}(\boldsymbol{x})), p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z})) = -\log p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{\phi}(\boldsymbol{x})) \quad (2.11)$$

From Eq. (2.11), the β-VAE loss function for a deterministic encoder can be expressed as [67],

$$\begin{aligned} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) &= E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\left[-\log p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \beta \log p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{z})\right] \\ &= -\log p_{\mathbf{x}|\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{\phi}(\boldsymbol{x})) - \beta \log p_{\mathbf{z},\boldsymbol{\theta}}(\boldsymbol{\phi}(\boldsymbol{x})). \end{aligned} \quad (2.12)$$

When $\beta = 1$, Eq. (2.12) corresponds to the VAE loss function in Eq (2.10). Setting $\beta \geq 1$ can impose a hard constraint on the latent representation [68] to limit the coding cost. In short, the approximated β-VAE loss is the sum of the reconstruction loss and the cost of encoding $\mathbf{z}$ weighted by β. The weight β facilitates a rate-distortion trade-off parameter. One extreme case would be to encode nothing, i.e., $\mathbf{z} = 0$, in which case the encoding cost is zero whereas the reconstruction loss from $\mathbf{z} = 0$ would be gigantic. The other extreme case would be to encode $\mathbf{x}$ directly, i.e., $\mathbf{z} = \mathbf{x}$, in which case the reconstruction loss is 0 whereas coding cost would be high.

Based on the β-VAE concepts for image reconstruction, in Chapter 5, we develop a VAE-based classifier for joint image compression and classification.

2.3 LINEAR DISCRIMINATIVE REPRESENTATION (LDR)

Modeling complex, non-linear real world data ($\boldsymbol{X}$) such as image or video essentially finds a deterministic function that transforms real world data into an intermediate (latent) representation. Such latent vectors corresponding to images of same class are expected to locate closer while those of different classes are expected to locate further in the learned latent domain of $\boldsymbol{Z}$. Transformation functions are preferred to be a subset of standard modeling functions such as Gaussian and transformed latent representations are to be linear and low dimensional [52]. Further, the latents needs to be sufficiently expressive to support the inverse transformation (reconstruction) of input data, and essentially discriminative to support multiple DL tasks such as classification, object detection etc. Recent "Linear Discriminative Representations" concept [53] explores one such family of low dimensional, linear latents.

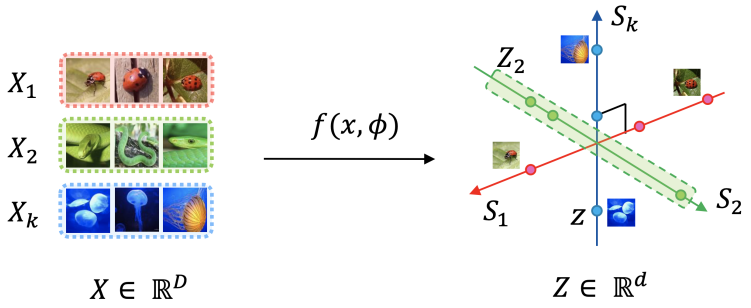


Figure 2.6: LDR maps images $\boldsymbol{X}$ in complex non-linear manifolds to latent representations $\boldsymbol{Z}$ distributed in linear orthogonal subspaces [53]. Images $\boldsymbol{X}_1$ of the same class: "lady bug" are mapped to the linear subspace $\boldsymbol{S}_1$.

Let the image set $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_m] \in \mathbb{R}^{D \times m}$ be m i.i.d. samples of dimension D. An encoder $f(\cdot, \boldsymbol{\phi})$ parameterized by $\boldsymbol{\phi}$ transforms each image sample $\boldsymbol{x}$ to a d-dimensional ($d < D$) learned representation $\boldsymbol{z}$ such that $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{\phi})$, which can be later used in image analysis tasks such as to predict the class label of the image $\boldsymbol{x}$ or image synthesis objectives to reconstruct the input image. We write the set of latent vectors mapped from set $\boldsymbol{X}$ as $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_m] \in \mathbb{R}^{d \times m}$.

$$\boldsymbol{x} \xrightarrow[f(\cdot,\boldsymbol{\phi})]{\text{Encoder}} \boldsymbol{z}(\boldsymbol{\phi}) \tag{2.13}$$

2.3.1 *Maximal Coding-Rate-Reduction (MCR^2)*

Recent work [53] has proposed the principle of MCR^2, which finds learned LDR $\boldsymbol{Z}(\boldsymbol{\phi})$ distributed in linear, class-wise orthogonal (decoupled) subspaces as shown in Fig. 2.6. Note that images in the same class are mapped to the same subspace and linearly distributed over that subspace.

The objective of MCR^2 principle [53] is to maximize the difference ΔR between the global code length and the summation of class-wise code lengths of latent features $\boldsymbol{Z}$, coded as Gaussian sources with distortion ϵ:

$$\Delta R(\boldsymbol{Z}|\epsilon,\boldsymbol{\Pi}) = \underbrace{\frac{1}{2}\log_2\det\left[\boldsymbol{I}+\alpha\boldsymbol{Z}\boldsymbol{Z}^\top\right]}_{R(\boldsymbol{Z},\epsilon)} - \underbrace{\sum_{j=1}^{k}\frac{\gamma_j}{2}\log_2\det\left[\boldsymbol{I}+\alpha_j\boldsymbol{Z}\boldsymbol{\Pi}_j\boldsymbol{Z}^\top\right]}_{R^c(\boldsymbol{Z}|\epsilon,\boldsymbol{\Pi})}, \tag{2.14}$$

where, $\alpha = d/m\epsilon^2, \alpha_j = d/\mathrm{tr}(\boldsymbol{\Pi}_j)\epsilon^2$ and $\gamma_j = \mathrm{tr}(\boldsymbol{\Pi}_j)/m$. The first term $R(\boldsymbol{Z}|\epsilon)$ in Eq. (2.14) measures the expected code length (rate) in bits of a sample from $\boldsymbol{Z}$. Given available class label information, the second term $R^c(\boldsymbol{Z}|\epsilon,\boldsymbol{\Pi})$ measures the average code length in bits when latents $\boldsymbol{Z}$ are distributed in k disjoint class-wise subspaces: $\boldsymbol{Z} = \boldsymbol{Z}_1 \cup \cdots \cup \boldsymbol{Z}_k$. The membership set $\boldsymbol{\Pi}$ of binary diagonal matrices $\{\boldsymbol{\Pi}_1, \cdots, \boldsymbol{\Pi}_k\} \in \mathbb{R}^{m\times m}$ with $\sum_{i=1}^{k}\boldsymbol{\Pi}_i = \boldsymbol{I}_m$ denotes the above partitioning of $\boldsymbol{Z}$ into k classes. In this setting, each partition consists of $\mathrm{tr}(\boldsymbol{\Pi}_j)$ samples [69].

Maximizing the first term of Eq. (2.14) induces latent vectors $\boldsymbol{z}$ in Fig. 2.6 to distribute as further as possible, whereas minimizing the second term pushes latent vectors of the same class closer. Jointly, maximizing Eq. 2.14 linearizes feature embedding $\boldsymbol{z}$ for each class to promote in-class compressibility and inter-class discriminability. Recent works have demonstrated promising benefits by exploiting the above characteristics of LDR for joint image classification and compression in networked AI settings [59, 70]. In particular, [59] proposes to improve rate-classification accuracy performance by transmitting LDR $\boldsymbol{z}$ with learned subspaces for

cloud inference. In Chapters 7-8 in this dissertation, we propose LDR-guided auto-encoders for improved rate-accuracy performance that are robust to extreme class imbalance scenarios during image classification.

Part I

IMPROVING CONVENTIONAL CODECS

3

FASTER AND ACCURATE DEEP LEARNING FOR JPEG2000 IMAGES

Image/video classification in a networked environment is among the top AI applications for years to come. Given its rapidly growing popularity and usage for visual applications, JPEG2000 is playing an increasingly vital role in cyber intensive and autonomous systems. In this chapter, we explore new and better ways to exploit the JPEG2000 (j2k) encoding in critical AI tasks such as image and video classification.

3.1 INTRODUCTION

In a wide variety of application scenarios involving low complexity IoT and networked sensors, traditional AI functionalities often rely on cloud computing to handle high complexity processing tasks such as image/video analysis. In cloud-based networked image analysis applications, neural networks (NNs) are trained at the cloud servers on the images transmitted by source devices. During inference of networked image classification task, the trained deep CNN classifier at the cloud predicts the class label for received input image. In such networked-AI setting, there is a clear trade-off among computation complexity, network payload, and performance in terms of accuracy.

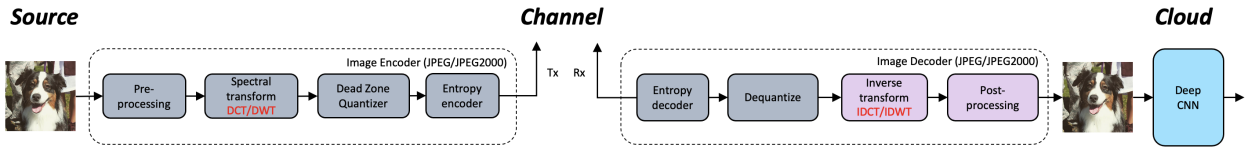


Figure 3.1: Conventional training and inference for cloud-based image classification. Pre/post processing involve level offset and color transformations as discussed in 2.1

In order to conserve limited channel bandwidth and storage capacity at the cloud, source devices often encode and compress the images before transmitting to the cloud by utilizing

standardized compression techniques such as JPEG2000. Since majority of NNs are designed and optimized to classify images in the spatial (RBG) domain, the cloud currently receives and decodes the compressed j2k images back into the RGB domain before forwarding them to pre-trained NNs for further processing, as illustrated in figure 3.1. However, given that such received images are analyzed by DL algorithms and not by human experts, a natural question arises is to how to achieve faster training and inference with improved accuracy in a cloud based image classification under bandwidth, storage and computation constraints.

3.1.1 *Related works: Image classification on DWT coefficients*

Image classification in spectral domain has been studied by the ML community over the past few years. The compact representation of the images in spectral domains promises faster classification [71–73]. As introduced in Chapter 2, JPEG2000 codec transforms spatial domain RGB images to spectral domain CDF 9/7 wavelets during lossy compression. Despite the considerable volume of works on the use of DWT coefficients for DL (e.g., [25, 73–76]), there is only a handful of published works that use CDF 9/7 wavelets for classification. For instance, the authors of [73–76] uses DB1, mostly known as 'Harr' wavelet for DWT calculation. The work [73] discusses the significance of each sub-band from DWT for image classification, which suggested a framework for applying the inference results of each sub-band for final classification decision to achieve lower misclassification rate. The authors of [74] proposed a DWT based approach for texture classification and suggested to improve image classification by stacking the sub-bands of DWT. By only using DB1 coefficients of the level-1 sub-band (LL_1) , the authors of [75] report a convolutional neural network (CNN) is 10 times faster for feature extraction and recognition of images.

3.1.2 *Is reconstruction necessary?*

The authors of [71] applied existing JPEG codec to directly extract the Discrete Cosine transformed (DCT) coefficients for classifying ImageNet [77] data set. They claim faster classification by reducing some blocks of the ResNet stack, together with the time saved from not reconstructing RGB images before ResNet. Similarly, [78] has shown that reconstruction of RGB images from custom compressed coefficients is unnecessary for classification purposes. Inline with the approach of [71] on JPEG, we use the standard JPEG2000 codec to extract DWT coefficients for classification and show that the conventional use of image reconstruction is unnecessary for JPEG2000 encoded classification by constructing and training a deep CNN model with the DWT coefficients transmitted in standard j2k stream. See figure 3.2. We show not only faster but also more accurate classification results even before considering the reconstruction savings based on the experiments on both CIFAR-10 [12] as well as Tiny ImageNet (a subset of ImageNet) data sets.

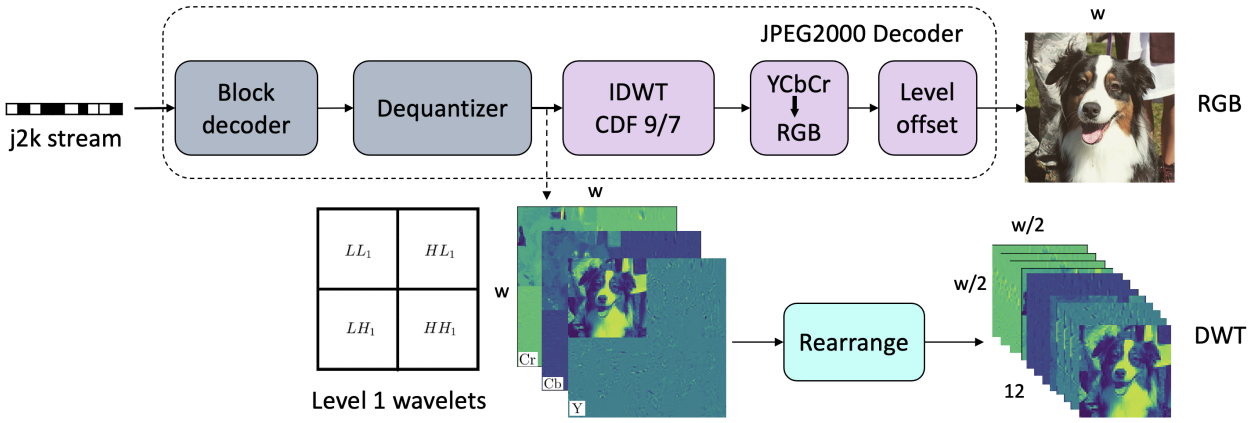


Figure 3.2: At the cloud during inference, decoder can skip IDWT, color transformation and level offset for faster classification. For illustration, we show a level-1 DWT compressed image. We rearrange the CDF 9/7 coefficients as inputs to the deep CNN.

To the best of our knowledge, this is the first reported attempt that successfully applies DWT coefficients extracted from within the JPEG2000 decoder for image classification. The authors of [78] demonstrated a similar concept of skipping reconstruction in ResNet classifier

for a less common convolution encoder/decoder instead of the widely popular JPEG2000 encoder/decoder in multimedia applications. On the other hand, none of the previous works addressed the extension of regular augmentation techniques in transform domains such as DCT or DWT. Furthermore in this chapter, we establish that more accurate classification is also possible by deploying shallower models to benefit from faster training and inference in comparison to models trained with spatial RGB image inputs.

The rest of this chapter is organized as the following. In Sec. 3.2, we introduce our proposal of DWT-specific image augmentation and shallow classifier models targeting JPEG2000 compressed images. Sec. 3.3 explores the consistency of the proposed solution over band limited channels at different compression ratios of the JPEG2000 codec. See Appendix B for the details of the model architectures, fine-tuning and additional classification experiments.

3.2 CLASSIFICATION OF JPEG2000 COMPRESSED IMAGES

The proposed decoder, as shown in Figure 3.2, extracts the CDF 9/7 coefficients using the dequantizing block which multiplies the received DWT coefficients with the same step size used by the encoder during quantization. In our experiments, we modified the open source 'C' codes of the OpenJPEG project[1] to extract DWT coefficients and to generate RGB images. Similar to [25], three level-1 DWT channels corresponding to Y, Cb, and Cr are stacked into a tensor of 12 sub bands each with the height and width of half of the RGB image.

3.2.1 *DWT-specific Image Augmentation*

Image augmentation is an essential step in deep CNN models to combat over fitting. By training with augmented images one can generalize the CNN model to classify unseen images during training. In each mini batch, augmentation can provide random transformations such

[1] http://www.openjpeg.org/

as horizontal flipping, rotation, vertical and horizontal shifting. These transformations are meaningful in the spatial domain like RGB or YCbCr. However, for inputs in DWT or other transform domains, these conventional transformations do not have physical meaning and have proven ineffective. Figure 3.3 shows a clear distortion in (c) as a result of incorrectly flipped high frequency sub-bands.

(a) original (b) flipped in RGB (c) fillped in DWT (d) proposed

Figure 3.3: Image augmentation in DWT domain. (a) Original image. 'Brownie' of size 112×112. (b) shows the horizontally flipped image in RGB domain. During training, mini batches are in DWT domain. (c) shows the result of using the same transformation in DWT domain and reconstructed in RGB for visualization. (d) horizontally flipped in DWT domain with proposed augmentation transform and reconstructed in RGB for visualization. (b) and (d) are exactly same proving the effectiveness of the proposed method. (c) is distorted implying the ineffectiveness of the conventional augmentation transforms.

We address this problem by proposing the following augmentation transforms. Let n be the image dimension. Define $X_s \in \mathcal{R}^{n \times n}$ and $X_{w_A} \in \mathcal{R}^{n \times n}$ where X_{w_A} is the DWT of X_s and A is the DWT matrix of a wavelet a. Now we can write X_{w_A} as separable column and row transformations (See [60] for more details).

$$X_{w_A} = A^T X_s A. \tag{3.1}$$

We define the regular spatial domain transformation H as

$$X_{s_H} = X_s H \tag{3.2}$$

where X_{s_H} is the augmented X_s in the spatial domain. We suggest an alternative transformation $\acute{H}$ defined as

$$\acute{H} = A^{-1}HA \tag{3.3}$$

which replaces H in augmentation in DWT domain. (See Section 2.1 for the formulation of A).

3.2.2 *Computation Speed and Accuracy*

We achieve faster training and inference in two ways. First, we can save computation cost by not reconstructing to achieve a "reconstruction gain". Second, shallower CNN classification models are sufficient for image classification in the transform domain, leading to efficient forward/backward propagation to achieve a "shallow model gain". By applying DWT-specific augmentation, we further improve classification accuracy. The first two methods are consistent with known results for JPEG as faster classification can be achieved with fewer convolution blocks [71].

Reconstruction Gain

Deep CNNs such as ResNet [15] thrive on RGB inputs and have been shown effective with augmentation transforms like flipping, random cropping and rotations etc. In cloud based image classification, the receiver node receives j2k stream of each inference image. We demonstrate that the JPEG decoder steps after the dequantizer as shown in Figure 3.2 are unnecessary to achieve the same level accuracy as reconstructed RGB images. Hence the computation required for IDWT, YCbCR to RGB conversion, and the level offset can be omitted. We call this computation-time saving, "reconstruction gain" which is dominated by the IDWT of CDF 9/7 wavelets in JPEG2000. The reconstruction gain amounts to around 20% decoding time of the OpenJPEG CPU implementation. See Table 3.1.

Shallow Models

For JPEG2000 encoded images, deep CNN models with fewer number of residual blocks are sufficient to achieve a given classification accuracy in DWT domain with CDF 9/7 wavelets in comparison to classification in the RGB domain. On the other hand, the width of the RGB inputs is 2 times larger than the CDF 9/7 inputs (for level-1 DWT), but DWT requires more convolution filters for each layer to compensate the larger input depth (12 channels) in CDF 9/7 compared to the 3 RGB channels. It can be shown that models with less number of residual blocks compared to models with more residual blocks having the same number of convolution operations perform faster training and inference. See section 3.3.1 for more details.

3.3 EXPERIMENTS AND RESULTS

In the initial experiments, we used CIFAR-10 data set which consists of 50,000 training images and 10,000 testing images of size 32×32 belonging to 10 classes. We then repeated the experiments on Tiny ImageNet, which is a subset of ImageNet data set [77]. Tiny ImageNet consists of RGB images of size 64 × 64 belonging to 200 classes, each class with 1300 training images and 50 validation images.

To compare the classification accuracy in RGB and DWT domains, we start by encoding original (source) images in the training set using JPEG2000 encoder at compression ratio r to generate j2k streams for each image. We then decode the j2k streams into RGB images

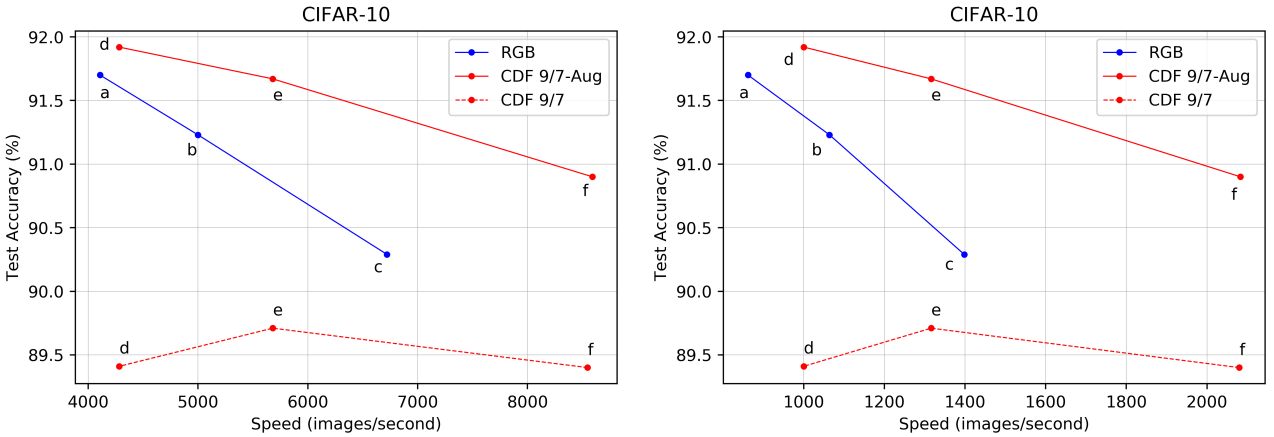
Table 3.1: Reconstruction time savings for different image sizes.

Image size	decoding time (ms)	recon. saving (ms)	recon. saving (%)
32×32	3.57	1.37	38.4
64×64	7.26	2.45	33.5
224×224	23.50	4.13	17.6

for RGB domain inputs and, harvest their DWT values inside the codec for DWT domain inputs as explained in Fig. 3.2.

Table 3.1 summarizes the reconstruction savings for processing different size images in DWT domain. We can see that the time saved by skipping RGB image reconstructing for inference and training accounts for around 20% of the total decoding time. The decoding time is based on an INTEL 7^{th} GEN CORE i7-7700HQ CPU.

Figure 3.4 compares the classification accuracy and the speed of inference and training for uncompressed images (i.e., $r = 0$) of CIFAR-10 data set. Each point in Fig 3.4 represents the test accuracy and the processing speed of a particular model. To generate the results for a method, we changed the number of ResNet blocks to obtain better accuracy at the cost of training and inference time. To calculate inference and training speed, we **did not include** the reconstruction savings as explained in Section 3.2.2. Inclusion of this extra gain would shift the DWT curves more favorably to the right. We gained approximately 2% improvement in classification accuracy by applying our proposed augmentation techniques to the CIFAR-10 data set.



(a) inference (b) training

Figure 3.4: (a). Test accuracy vs inference speed for the CIFAR-10 data set. Blue: using reconstructed RGB images, solid red: using DWT coefficients extracted from JPEG2000 codec with our proposed augmentation, dashed red: using DWT coefficients with regular augmentation. (b). Test error vs training speed/epoch. Rate is the number of images that go through the model/second.

Table 3.2 and Fig. 3.5 describe the ResNet model settings corresponding to the result points a, b, c, d, e and f in Fig. 3.4 and 3.7 for the CIFAR-10 data set. Note that the numbers of convolution layers of the models used for DWT are lower than those used for RGB. Although DWT models used more kernels requiring more parameters, the RGB input width is 2 times larger and evens out the time spent by DWT models on more kernels. We used an initial learning rate of 0.001 which is reduced progressively by 1/10 at 80, 120 and 160 over 200 epochs.

Table 3.2: Parameters for the ResNet models for RGB and CDF 9/7 inputs of CIFAR-10. w: input width, n_c: no of input channels, n_b: no of ResNet blocks per layer and k: average pooling size. See Fig. 3.5 for an illustration of model 'd'.

Parameter/Domain	RGB			CDF 9/7		
Model	a	b	c	d	e	f
w	32	32	32	16	16	16
n_c	3	3	3	12	12	12
n_b	4	3	2	3	2	1
nf_0	16	16	16	64	64	64
nf_1	16	16	16	64	64	64
nf_2	32	32	32	96	96	96
nf_3	64	64	64	144	144	144
k	8	8	8	4	4	4
no of CONV layers	27	21	15	21	15	9
no of parameters (M)	0.37	0.27	0.18	1.79	1.17	0.55

For ImageNet experiments, we used a ResNet with bottleneck architecture described in [15] for both RGB and DWT domains. This resulted in 3% accuracy gain over basic ResNet architecture for the DWT domain and 0.3% improvement for the RGB domain. The initial learning rate is 0.001 which is reduced by 1/10 at 30, 60 and 90 over 95 epochs, respectively. For both CIFAR-10 and Tiny ImageNet, we used 'Adam' as the optimizer.

We summarize the best classification models obtained for CIFAR-10 and Tiny ImageNet data sets in Table 3.3 and Table 3.4, respectively. We repeated each experiment 3 times to compute the average and standard deviation of accuracy rates. Both data sets validate our claim of faster training and inference for JPEG2000 compressed images even without

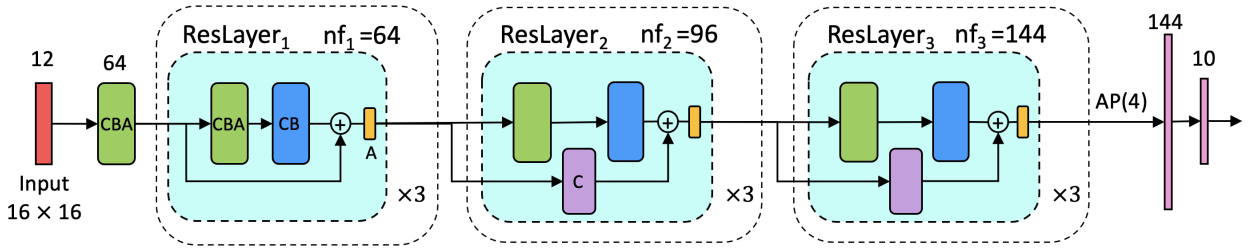


Figure 3.5: Modified Resnet for model 'd'. Same colored blocks are identical. Each block of ResLayer$_i$ consists of nf_i convolution filters.

considering the reconstruction saving which is around 20% of the total decoding time. Similarly, both data sets demonstrate the effectiveness of our proposed augmentation techniques for DWT domain images. CIFAR-10 shows over 2.5% and Tiny ImageNet achieves over 1% and 1.5% of top-5 and top-1 accuracy improvement, respectively.

Table 3.3: Results of CIFAR-10 based on models 'a' and 'd'.

parameter	RGB	CDF 9/7Aug (ours)	CDF 9/7
Test Acc. (%)	91.70(±0.07)	**91.92**(±0.11)	89.41(±0.07)
No of Conv. layers	27	21	21
Training rate/epoch (images/s)	862	**1000**	1000
Inference rate (images/s)	4108	**4283**	4259

Table 3.4: Results of Tiny ImageNet.

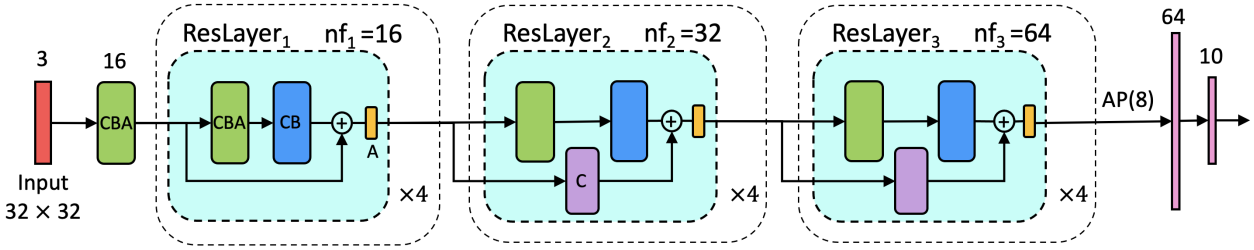
parameter	RGB	CDF 9/7Aug (ours)	CDF 9/7
Top 5 test Acc. (%)	89.06(±0.03)	**89.08**(±0.02)	87.92(±0.07)
Top 1 test Acc. (%)	67.35(±0.11)	**67.56**(±0.09)	65.78(±0.36)
No of Conv. layers	40	31	31
Training rate/epoch (images/s)	670	**694**	694
Inference rate (images/s)	1865	**1881**	1881

When compared to classification accuracy in the RGB domain, Both CIFAR-10 and Tiny Imagenet results show top-1 accuracy improvement of 0.2% on average. Overall, both experiments suggest that RGB image reconstruction from j2k stream is not necessary for CNN classification. Inclusion of the reconstruction saving, DWT domain models can perform more than ×2 times faster over RBG domain. Although faster decoders implemented on

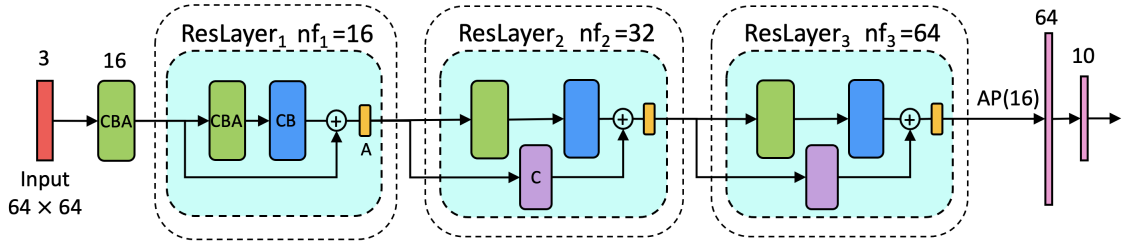
GPUs may help improve the computation speed, as shown in Table 3.1, this computation gain amounts to around 1/5 of the decoding time.

3.3.1 *Shallow models are faster!*

We show that the models with less number of ResNet blocks are faster in training and inference compared to the models with more number of blocks that uses similar number of convolution operations. For instance, compare models layer-9 and layer-27 ResNets. See Fig. 3.6. Both models take around 6.4M convolution operations. Model (a) has a 893 images/second training rate and model (b) has a 1250 images/second training rate. This is because forward and backward propagation for same number of convolution operations is efficient for shallow models. This helps shallow ResNet models with DWT inputs to perform faster even with more parameters compared to RGB models.



(a) layer-27 ResNet

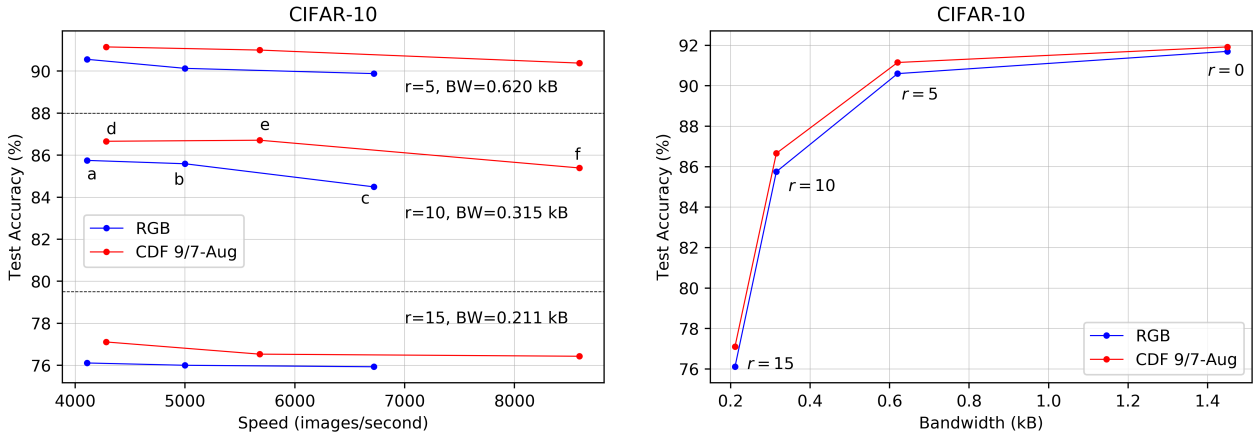


(b) layer-9 ResNet

Figure 3.6: Both models take around 6.4M convolution operations. Model (a) has a 893 images/second training rate and model (b) has a 1250 images/second training rate.

3.3.2 *Experiments over Bandwidth Constrained Channels*

Figure 3.7 shows how the classification accuracy behaves under bandlimited channels. Results in Fig. 3.7(a) confirm faster and accurate classification for DWT domain models for channels with limited bandwidth. Results given by Fig. 3.7(b) show the change of accuracy for a particular model for different bandwidth constraints (compression ratio r). These results for JPEG2000 encoder that uses CDF 9/7 DWT coefficients are consistent with the tests results from DB1 wavelets given in [25].



(a) (b)

Figure 3.7: (a). Models designed for CDF 9/7 DWT coefficients using the proposed augmentations are faster and more accurate under limited bandwidth. Parameter 'r' in JPEG2000 codec adjusts compression ratio and 'BW' is the bandwidth in terms of average image size. (b). Effect of channel bandwidth. RGB uses model 'a' and CDF 9/7 uses model 'd'. This result is consistent with result from DB1 wavelets in [25]

3.4 CONCLUSIONS

This chapter investigated cloud-based deep CNN image classification in congested communication networks. We proposed to directly train deep CNN classifier for JPEG2000 encoded images by using its DWT coefficients in j2k streams and achieved better accuracy and faster

computation by using shallower CNNs in the DWT domain. We further introduced new augmentation transforms to develop CNN models that are robust to common communication bandwidth constraints in cloud based AI applications. Next chapter explores the optimization of quantization of JPEG2000 codec based on deep learning for improved rate-accuracy performance.

4

JOINT OPTIMIZATION OF CONVENTIONAL J2K COMPRESSION AND DEEP LEARNING

In chapter 3, we discussed how to utilize JPEG2000 compressed images for faster and accurate classification. In this chapter, we explore how to optimize image compression codecs like JPEG2000 for deep learning in transformed domains such as DWT. We propose an end to end joint compression and learning network to fulfill this task and show significant improvements in bandwidth savings for a given classification accuracy and vise-versa.

4.1 DEEP LEARNING OVER BANDWIDTH CONSTRAINED CHANNELS

In cloud-based DL scenarios, where model is trained at the cloud and accepts query images for inference, network bandwidth is often limited and costly. Thus, it is important to encode and compress images efficiently before transmitting to a high performance classifier node aiming to achieve high inference accuracy without consuming excessive network bandwidth. In lossy image codecs like JPEG/JPEG2000, losing information during compression primarily happens in the quantization process. See figure 4.1.

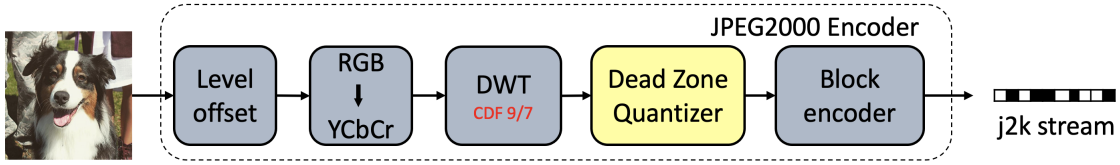


Figure 4.1: JPEG2000 lossy encoding process. Losing information during compression primarily happens at the "dead zone quantizer".

Quantization maps a range of values with floating point precision to an integer. JPEG2000 follows the following definition of the uniform deadzone quantizer [29]

$$q(x) = \text{sgn}(x)\left\lfloor \frac{|x|}{d_s} \right\rfloor. \tag{4.1}$$

Here, $q(x)$ is the quantization index for sub-band coefficient x whereas d_s is quantization step for sub-band s. $\lfloor \cdot \rfloor$ indicates the "floor" operation. By making d_s larger, we can make the required bandwidth smaller at the cost of precision of the reconstructed (dequantized) values. In JPEG2000 [29], d_s is defined as follows:

$$d_s = \begin{cases} \tau/2^l & \text{if } s = \text{LL}_l \text{ (approximated image)} \\ \tau/2^{k-1} & \text{if } s = \text{LH}_k, \text{HL}_k \text{ where } \quad k = 1, \cdots, l \\ \tau/2^{k-2} & \text{if } s = \text{HH}_k \text{ where } \quad k = 1, \cdots, l. \end{cases} \tag{4.2}$$

with parameter τ is determined by

$$\tau = 2^{R-c+l}\left(1 + 2^{-11} \cdot f\right),$$

in which l is the number of levels of DWT, R is the number of bits to represent original pixel values, while c and f denote the number of exponent bits and mantissa bits, respectively.

To examine the connection between image bandwidth and classification accuracy we can incrementally set the compression ratio r of the JPEG2000 codec and train a deep CNN for each set of images compressed at r. Under this setting, in the previous chapter, we observed that the classification accuracy significantly decreases with low bandwidth. Figure 3.7(b) shows the rate-accuracy ($\mathcal{R}$-$\mathcal{C}$) performance curve for CIFAR-10 data set. Data rate (bandwidth) is calculated in kB.

The slope of the $\mathcal{R}$-$\mathcal{C}$ curve depends on the quantization step sizes used in JPEG2000 codec. Image codecs like JPEG/JPEG2000 use manually defined quantization step sizes for different compression ratios. See eqn (4.2). But these step sizes have been optimized to generate best visual quality for a given data rate. In other words, these quantization step sizes are optimized to deliver better rate-distortion ($\mathcal{R}$-$\mathcal{D}$) performance. So these pre-defined values are not optimal for deep learning tasks that uses transformed domain inputs as DWT.

On the other hand, these optimal step sizes are task dependent. For instance, optimal d_s for image classification may not be optimal for image segmentation.

Thus, one important problem we wish to solve is to maintain a certain level of accuracy while minimizing the bandwidth consumption by optimizing the quantization parameters for transformed image sub-bands and coefficients for a given deep learning task. There is only a handful of works that discuss the effects of image compression and the optimized representation of transformed coefficients for bandwidth efficiency for deep learning. The authors of [79] manually selected a quantization matrix for JPEG images that achieved similar detection accuracy with reduced bandwidth. This result demonstrates that current image encoding and quantization settings are not optimized for ML purposes. Furthermore authors of [80] considered an alternative compression method called DeepN-JPEG. By analyzing frequency components of images they suggested a manual quantization setting to control bandwidth.

In this chapter, we investigate effective ways to improve $\mathcal{R}$-$\mathcal{C}$ performance by reducing bandwidth of JPEG2000 images, while preserving high classification accuracy. Towards this goal, we identify the important image components and sub-bands for cloud based image processing. More importantly, we solve problem of how to autonomously quantize DWT coefficients for a target level of classification accuracy and propose a novel end to end, joint learning framework to optimize the quantization parameters for JPGE2000 encoded images.

We organize this chapter as the following. In Sec. 4.1.1, we briefly discuss the related works to the joint learning proposal. In Sec. 4.2, we present our proposed framework: "Quannet" that learns quantization step sizes of a given compression codec for a given DL task. Sec. 4.3 shows the experiment results on CIFAR-10 data set. In Sec. 4.3, we further discuss the operating principles of QuanNet. See Appendix B for details of the NN architectures used in the experiments.

4.1.1 *Related works*

The idea of joint learning has been presented for different objectives in the literature. For instance, a joint learning model for visual denoising and hand written digits classification was developed in [81]. For joint blurring, denoising and classification, a differentiable learning architecture was proposed in [82]. The novel aspect of our framework is the joint optimization of the deep learning network coupled with the optimization for image quantization to achieve the dual objectives of bandwidth compression and accurate inference.

For image classification, the role and importance of each sub-band is different [73]. Furthermore, coefficients within each sub-band also have different importance. This presents an opportunity to compress some information for transmission by allocating less bandwidth to less important sub-bands/coefficients with little performance loss. Similar argument is valid for image visualization purposes [29].

In communication networks serving cloud based machine learning tasks such as image recognition and classification, the source data needs to be transmitted from sensing nodes to more powerful network nodes with substantial computation and energy capacity. The transmission of such data consume both power and bandwidth. For this reason, it is important to develop efficient image compression for transmission over limited bandwidth while preserving a high degree of accuracy for cloud based classification.

4.2 QUANNET: JOINT QUANTIZATION AND DEEP LEARNING

We now propose an end-to-end joint learning architecture "QuanNet", aimed at jointly optimizing both the JPEG2000 encoder quantization and a given deep learning model to achieve efficient image encoding and high accuracy inference. For instance deep learning model can be a ResNet-18 for image classification in which case QuanNet jointly optimizes quantization for image classification or a Fully Convolutional Network (FCN) [22] for image

segmentation in which case QuanNet jointly optimizes quantization for image segmentation. Let's assume the given deep learning model is an image classifier in this case.

QuanNet shown in Fig. 4.2 consists of a quantization block, 'Quan' followed by a deep CNN typically trained for image classification. The Quan block (see Fig.4.3), parameterized by a trainable weight vector $\mathbf{w}$ is to quantize the DWT coefficients X of an image. Its output $X^q_{\mathbf{w}}$ is quantized according to the quantization parameters $\mathbf{w}$ before the deep CNN for classification.

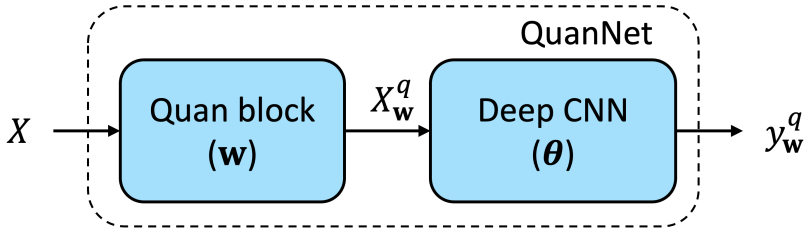


Figure 4.2: QuanNet consists of a quantization block followed by a deep CNN. $y^q_{\mathbf{w}}$ is the predicted label based on $X^q_{\mathbf{w}}$.

In this architecture, we focus on finding a uniform quantization interval $d_s = 1/w_s$ with $w_s \in [0, 1]$ for sub band s. For the ease of understanding (4.1) can be rewritten as follows:

$$q(x) = \text{sgn}(x)\left\lfloor \mathbf{w}_s|x| \right\rfloor. \tag{4.3}$$

To be able to learn quantization intervals parameterized by $\mathbf{w}$, Quan block should reproduce the functionality of (4.3). As illustrated in Fig. 4.3, the Quan block can be easily implemented as a set of 12 element-wise multiplication filters which has only 12 parameters followed by an approximated floor function. The floor function can be thought as an activation function acts on each output from the element-wise multiplication filters.

Since the floor function has zero gradient almost everywhere, one apparent obstacle is the possibility of slow convergence in training of QuanNet via back propagation. Similar to [83], to overcome this problem, we propose an approximation for the floor function, $\tilde{\text{flr}}(\cdot)$,

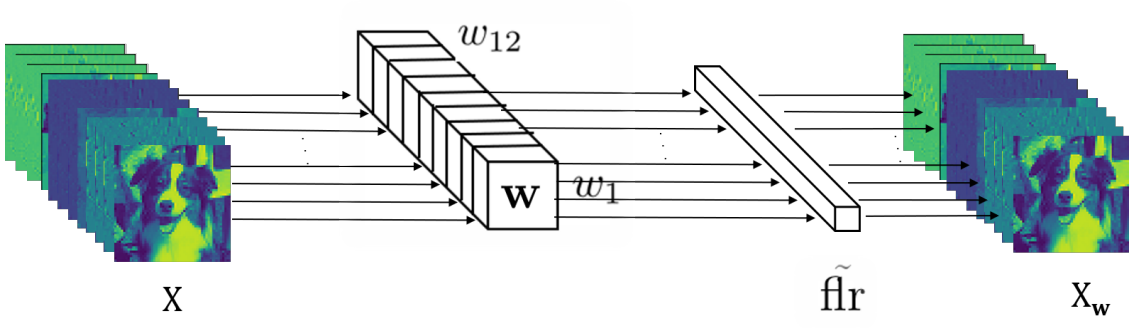


Figure 4.3: Illustration of Quan block: DWT coefficients are scaled by w_s before rounding down to nearest integers.

which is differentiable and easy to implement. One simple candidate is a finite summation of sigmoid functions with controllable parameters s:

$$\tilde{\text{flr}}(x) = \sum_{i=-N, i\neq 0}^{N} \text{sigmoid}\left[s(x-i)\right] - N. \tag{4.4}$$

This function generates integer output almost everywhere except for exact integer inputs. The number N depends on the range of the input x. A good approximation requires N to be chosen such that all DWT coefficients fall within $(-N, N)$. For instance figure 4.4 shows a dead zone quantization approximation for $N = 5$ and $s = 100$.

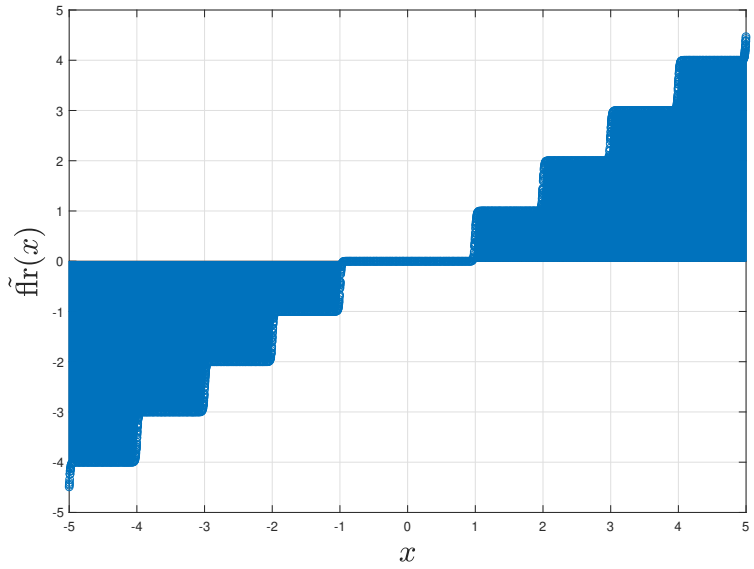


Figure 4.4: A differentiable dead zone quantization approximation for $N = 5$ and $s = 100$.

We define the loss function of QuanNet,

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}[\mathcal{L}_{\mathrm{CL}}(\boldsymbol{\theta}(X^q_{\mathbf{w}}), y_{\mathrm{gt}})] + \lambda \mathcal{L}_{\mathrm{quan}}(\mathbf{w}), \tag{4.5}$$

as a combination of traditional classification cross entropy loss $\mathcal{L}_{\mathrm{CL}}$ and a quantization loss $\mathcal{L}_{\mathrm{quan}}$. The first term of (4.5) averages the $\mathcal{L}_{\mathrm{CL}}$ over the training set: DWT of image and ground truth class label pairs $\{X, y_{\mathrm{gt}}\}$. The purpose of the second term $\mathcal{L}_{\mathrm{quan}}$ is to reduce the bandwidth required of the quantizer output. One simplest function for this job is a regularizer such as,

ℓ_1-regularizer,

$$\mathcal{L}_{\mathrm{quan}}(\mathbf{w}) = |\mathbf{w}|_1 \tag{4.6}$$

ℓ_2-regularizer,

$$\mathcal{L}_{\mathrm{quan}}(\mathbf{w}) = \|\mathbf{w}\|^2 \tag{4.7}$$

or a log regularizer in the form,

$$\mathcal{L}_{\mathrm{quan}}(\mathbf{w}) = \sum_{s=1}^{12} \log(1 + w_s). \tag{4.8}$$

A function in this form can induce smaller values of $w_s > 0$, thereby leading to lower bandwidth requirement for the quantized coefficients. By regulating λ, different emphasis on bandwidth can be enforced.

The training objective is to find the optimum vector $\mathbf{w}_o$ such that,

$$\mathbf{w}_o = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}|\lambda). \tag{4.9}$$

By training QuanNet, we can optimize a vector of quantization parameters $\mathbf{w}$ that can quantize DWT coefficients while maintaining the awareness of classification accuracy. Smaller values of w_s favor $\mathcal{L}_{quan}$ but may lead to larger $\mathcal{L}_{\mathrm{CL}}$ at the same time. With training, the network converges to a setting which gives a considerable classification accuracy under a

desired bandwidth level controlled by λ. Hence, for a given set of training images, QuanNet should generate an optimized set of quantization intervals $\mathbf{w}_o$ such that the quantized DWT coefficients can achieve higher $\mathcal{R}$-$\mathcal{C}$ performance. As a result, QuanNet allows more effective control of transmission bandwidth for cloud based image classification when compared with the quantization method used in conventional JPEG2000.

After learning optimal $\mathbf{w}_o$, we can replace the corresponding quantization step sizes d_s in the commercially used JPEG2000 encoder and use the trained deep CNN ($\boldsymbol{\theta}_o$) as the classifier. Figure 4.5 summarizes the proposed approach.

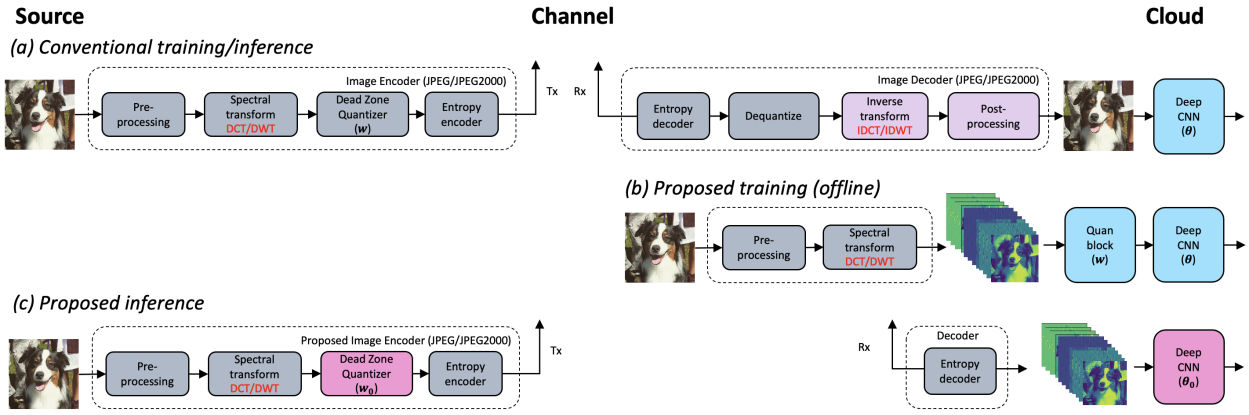


Figure 4.5: Proposed cloud-based training and inference with QuanNet

4.3 EXPERIMENT RESULTS

In this section, we provide the details of experiments related to joint quantization and deep learning. We experimented with CIFAR-10 data set with JPEG2000 encoded images at level 1 resolution using CDF 9/7 and DB1 wavelets. See section 2.1.1 for more details of how to form DWT coefficients using DB1 wavelets. We focused the initial experiments on DB1 coefficients rather than CDF 9/7 coefficients since substantial amount of work related to image classification with DWT coefficients has used DB1 wavelets compared to CDF 9/7 wavelets [73–76].

For RGB/YCbCr inputs we used models explained in Table B.1 and for experiments with DB1 wavelets, we used the ResNet models in Table B.5 as the deep CNN for joint quantization and image classification. We applied real time augmentation on the training images for both RGB and DB1 cases. During augmentation, we worked with batches of 32 training images. With ResNet-8, we used an initial learning rate of 0.001 which is reduced each time by 1/10 at 30, 60 and 90 over 100 epochs for both RGB and DB1 inputs. And for ResNet-20, we used the same initial learning rate with 1/10 reduction each time at 80, 120 and 160 over 200 epochs. We used the optimizer "Adam" both cases.

4.3.1 *Truncation with DB1 wavelets*

Exploring ways to reduce bandwidth for a given image, we experimented with DWT truncation. Truncation simply removes insignificant coefficients from the image representation during classification. One simple way is to replace a fraction of DWT coefficients or those below a given threshold by zeros. The classification results with ResNet-20 on CIFAR 10 after truncation of Level 1 - DB1 wavelets coefficients are given in Table 4.1. Based on our experimental results of Table 4.1, 50% of the DWT inputs can be truncated with little loss of classification accuracy.

Table 4.1: Effects of truncation (% of zeros) on classification.

Percentage of zeros	Test accuracy
4% (no truncation)	91.44%
50%	91.5%
75%	88.92%

Note that despite the truncation of DWT coefficients by 50%, the bandwidth reduction depends on how the remaining coefficients are encoded. For effective bandwidth reduction, the remaining DWT coefficients must be efficiently encoded with fewer bits without loss of performance.

4.3.2 *Quantization with DB1 wavelets*

Since DB1 is not heavily used in commercially available image compression codecs like JPEG2000, we started with the quantization equation given in eqn (4.2) and used parameters $R = 10$, $l = 1$, $c = 8$ and $f = 8$, which lead to step sizes of $d_{\mathrm{LL}_1} = 4.0156$, $d_{\mathrm{LH}_1} = d_{\mathrm{HL}_1} = 8.0313$ and $d_{\mathrm{HH}_1} = 16.0625$. Starting with these step sizes we doubled the step sizes each time and generated 4 points c_1, c_2, c_3 and c_4. Each of these points is equivalent to a compression ratio r as we discussed in section 4.1.

After the above quantization, we reconstructed YCbCr and RGB images from DB1 coefficients generated from with and without YCbCr conversion during encoding, to observe the compressibility of YCbCr color transformation. The test accuracy of image classification with ResNet-20 on CIFAR-10 is given in Table 4.2. We used compression ratio c_1 for quantization and note the higher percentage of zeros of the quantized DB1 coefficients results with YCbCr conversion and the accuracy loss is less than 0.5%. This observation is consistent with image classification with YCbCr inputs on ImageNet given in [71].

Table 4.2: Effects of quantized wavelet coefficients: Higher percentage of zeros implies higher compressibility of YCbCr

Representation	Percentage of zeros	Test accuracy
RGB w/o YCbCr	4% (no quantization)	91.44%
RGB w/o YCbCr	45% (c_1)	91.49%
YCbCr	62% (c_1)	91.01%

Figure 4.6 compares how the $\mathcal{R}$-$\mathcal{C}$ performance of DB1 coefficients and reconstructed RGB images change with limited bandwidth. We used arithmetic encoding to calculate the data rate (bandwidth) of DB1 coefficients for each image and averaged them to get the bandwidth for each compression ratio on CIFAR-10 data set. We observe that the classification accuracy degrades with limited bandwidth similar to JPEG2000 compressed CDF 9/7 coefficients shown in figure 3.7(b).

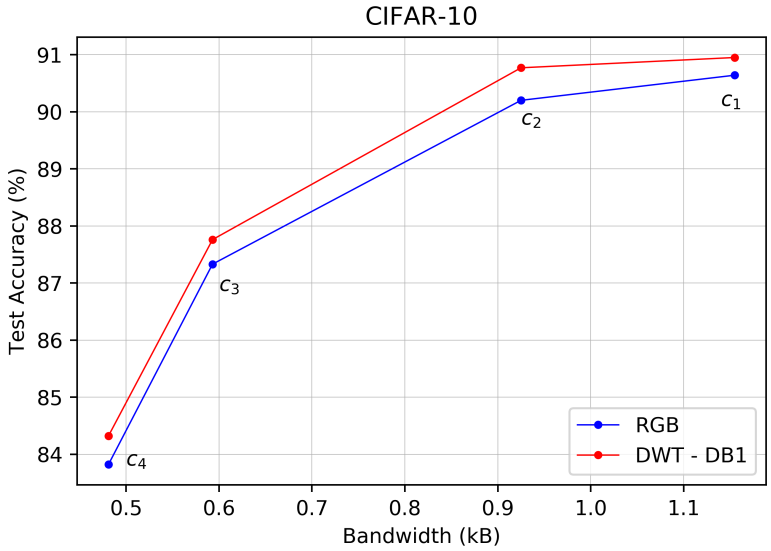


Figure 4.6: Classification accuracy for DB1 DWT coefficients degrades with limited bandwidth.

Then we applied the proposed QuanBlock for the same deep CNN models ResNet-20 and ResNet-8 given in Appendix B.5 to find the optimal quantization step sizes for given bandwidth requirements. For the experiments with DB1 coefficients we used ℓ_2 regularizer given in eqn (4.7) as the quantization loss function. Before training, we initialized the quntization weights vector $\mathbf{w}$ to ones. During training we used the same hyper parameters and number of epochs as we used for RGB inputs. Figure 4.7 compares the accuracy using QuanNet and RGB inputs that use pre-defined step sizes for quantization.

In figure 4.7, each point on the QuanNet (red) curves corresponds to a specific λ value in Eq. (4.5). By increasing this value, we can constrain the Quan block to produce images with lower bandwidth. This $\mathcal{R}$-$\mathcal{C}$ plot can be interpreted as an inverse rate-distortion characteristic curve drawn for a compression algorithm. But in this case instead of visual quality of the resulting images, we are interested in the inference accuracy of the learning task.

A few interesting observations are worth discussing. The rate of the classification accuracy drop of the QuanNet curve compared to RGB implies that jointly learning quantization step sizes for a specific task is beneficial to preserve accuracy under limited bandwidth. For instance, if we compare the points c_3 and $\lambda = 0.1$ of ResNet-20 curves, we can see

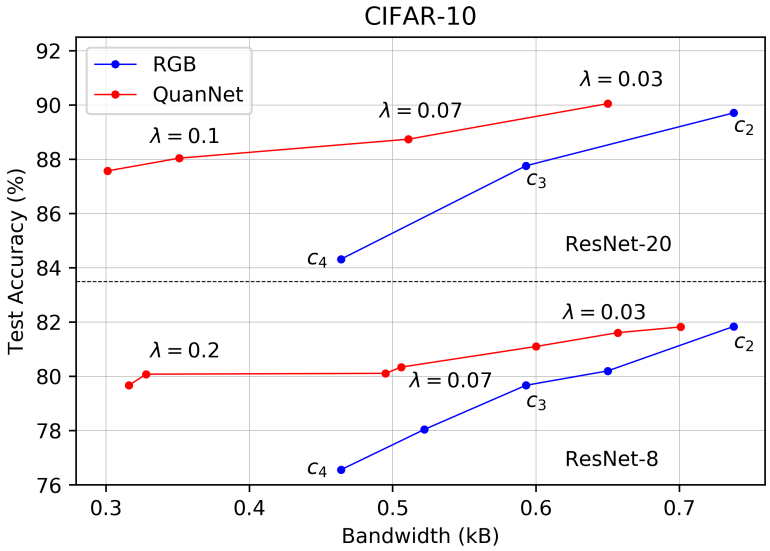


Figure 4.7: Accuracy and bandwidth comparison between QuanNet and baseline using ResNet-8 and ResNet-20. DB1 wavelet is used for DWT.

that accuracy of 88% can be achieved with 0.25kB less bandwidth which is about 40% of bandwidth savings. On the other hand, we can gain over 4% accuracy at 0.47kB bandwidth (point c_4).

Further more, QuanNet learns to drop insignificant sub-bands completely from CNN inputs instead of keeping sub-bands with very few nonzero DWT coefficients at lower bandwidths. For the different values of λ tested, QuanNet learned autonomously that it should transmit neither the HH sub-bands of all three channels, nor the HL and LH sub-bands of Cr and Cb channels. We further observed that QuanNet can learn to drop less information from important sub-bands. For example, consider the learned weight vector $\mathbf{w}_o$ corresponding to $\lambda = 0.07$ and $\lambda = 0.2$ for Resnet-8 case. The zero sub-bands are omitted in Table 4.3.

Table 4.3: Weights w_s learned from QuanNet for ResNet-8.

Sub-band (s)	w_s (for $\lambda = 0.07$)	w_s (for $\lambda = 0.2$)
Y_{LL}	0.0746	0.0879
Y_{HL}	0.0519	0
Cr_{LL}	0.0500	0.0803
Cb_{LL}	0.0710	0.0334

A few important results are clear from Table 4.3. First, the proposed QuanNet learning can autonomously determine that the Y channel of the images has information more important to classification instead of other 2 channels, a fact that is well known [71] from the visual perception perspective. Thus, the DWT sub-band coefficients from the Y channel are given larger weight (i.e., finer quantization). Secondly, as we change the learning emphasis by changing λ from 0.07 to 0.2, the QuanNet learns to drop more high frequency sub-bands, while at the same time it learns to transmit more information from Y_{LL} and Cr_{LL} by taking advantage of the bandwidth savings by dropping the high frequency sub-bands.

4.3.3 *Quantization with CDF 9/7 wavelets*

Unlike DB1 DWT coefficients, CDF 9/7 coefficients are highly compressible and heavily used in commercial compression codecs like JPEG2000. In this section we validate the performance of the QuanNet by comparing to JPEG2000 compression performance. To observe how the limited bandwidth affects the classification accuracy, we changed the compression ratio r of the j2k encoder and extracted CDF 9/7 DWT coefficients and RGB inputs at the decoder. Following the same pattern as DB1, the classification accuracy drops significantly at the lower bandwidths. See figure 3.7(b).

To use as the inputs for QuanNet, we extracted the unquantized CDF 9/7 coefficients before the quantization block of the encoder (see Fig. 4.1). For RGB inputs we used model 'b' and for experiments with CDF 9/7 coefficients, we used the model 'd' explained in Table 3.2 as the deep CNN for joint quantization and image classification. We applied real time augmentation on the training images for both RGB and CDF 9/7 cases. During augmentation, we worked with batches of 32 training images. We used an initial learning rate of 0.001 which is reduced each time by 1/10 at 80, 120 and 160 over 200 epochs for both RGB and CDF 9/7 inputs for both models. We used the optimizer "Adam" both cases.

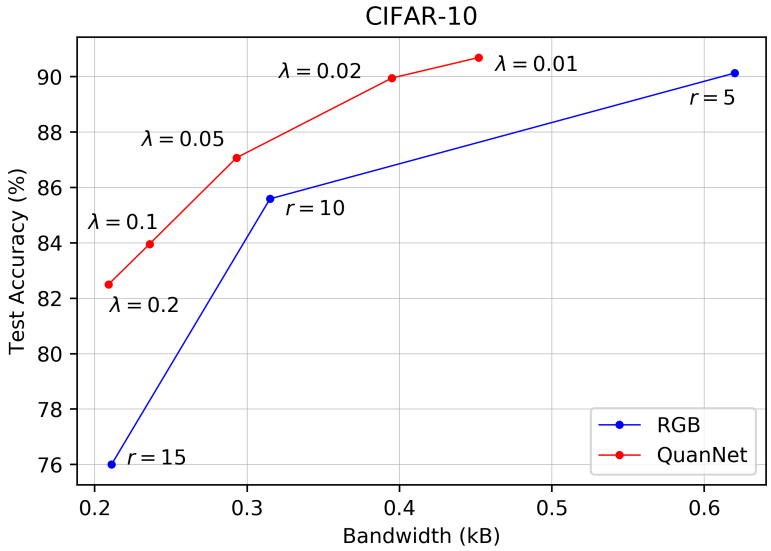


Figure 4.8: Accuracy and bandwidth comparison between QuanNet and baseline using model 'b' and 'e' respectively. CDF 9/7 wavelet is used for DWT.

We used log loss as in Eq. (4.8) as the quantization loss. Before training, we initialized the quantization weights vector $\mathbf{w}$ to ones. After learning the quantization step sizes for each λ value, we replaced the step sizes of the JPEG2000 codec with the learned values and calculated the average bandwidth after encoding CIFAR-10 data set with the new encoder. Fig. 4.8 illustrates the accuracy and bandwidth gains of QuanNet. These improvements are consistent with DB1 coefficients given in Fig. 4.7. We can see an accuracy improvement of over 6% at the bandwidth of 0.21 kB.

4.4 DISCUSSION

To further understand how QuanNet chooses to reduce bandwidth by completely dropping some subbands we summarize the all non zero sub bands for both RGB and QuanNet cases in Table 4.4.

There are more all zero sub bands learned from QuanNet compared to JPEG2000 for lower compression ratios $r = 5$ and $r = 10$ implying higher compressibility gained from joint

Table 4.4: QuanNet learned sub bands of CDF 9/7 coefficients.

	RGB		QuanNet
comp. ratio (r)	Sub bands	λ	Sub bands
5	$\begin{bmatrix} Y_{\text{LL}}, Y_{\text{HL}}, Y_{\text{LH}}, Y_{\text{HH}} \\ Cb_{\text{LL}}, Cb_{\text{HL}}, Cb_{\text{LH}}, - \\ Cr_{\text{LL}}, -, -, - \end{bmatrix}$	0.01	$\begin{bmatrix} Y_{\text{LL}}, Y_{\text{HL}}, Y_{\text{LH}}, - \\ Cb_{\text{LL}}, -, -, - \\ Cr_{\text{LL}}, -, -, - \end{bmatrix}$
10	$\begin{bmatrix} Y_{\text{LL}}, Y_{\text{HL}}, Y_{\text{LH}}, - \\ Cb_{\text{LL}}, -, -, - \\ Cr_{\text{LL}}, -, -, - \end{bmatrix}$	0.05	$\begin{bmatrix} Y_{\text{LL}}, -, -, - \\ Cb_{\text{LL}}, -, -, - \\ Cr_{\text{LL}}, -, -, - \end{bmatrix}$
15	$\begin{bmatrix} Y_{\text{LL}}, -, -, - \\ Cb_{\text{LL}}, -, -, - \\ -, -, -, - \end{bmatrix}$	0.2	$\begin{bmatrix} Y_{\text{LL}}, -, -, - \\ Cb_{\text{LL}}, -, -, - \\ -, -, -, - \end{bmatrix}$

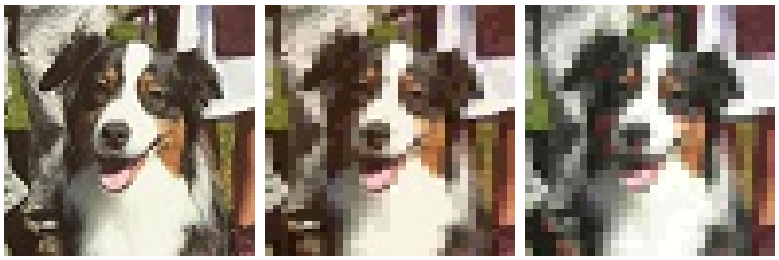
learning. For lower bandwidths at $r = 5$ and $\lambda = 0.2$ have same non zero bands and similar bandwidth, but QuanNet learns to classify over 6% better under joint learning.

To validate the applicability of the proposed method, we saved both learned quantization parameters $\mathbf{w}_o$ and $\boldsymbol{\theta}_o$ after training. Then we used exact floor function during quantization using $\mathbf{w}_o$ and observed same test accuracy with $\boldsymbol{\theta}_o$. This proves that the approximated floor function we used during training served its purpose well. On the other hand, we observed that training a new deep CNN with learned $\mathbf{w}_o$ to quantize cannot converge to same test loss as joint learning.

$$\mathcal{L}_{\text{CL:joint}}(\boldsymbol{\theta}_o(X^q_{\mathbf{w}_o}), y_{\text{gt}}) < \mathcal{L}_c(\boldsymbol{\theta}_t(X^q_{\mathbf{w}_f}), y_{\text{gt}}) \tag{4.10}$$

Here $\boldsymbol{\theta}_t$ is the deep CNN model that converges by training with fixed $\mathbf{w}_f$. This observation implies the effectiveness of the joint learning concept. By allowing a wiggle room for quantization parameters during training, parameters of the classifier can also converge to a better point compared to fixed quantization case.

Moreover, the reconstructed images from DWT coefficients at $\mathcal{R}$-$\mathcal{C}$ points corresponding to $\lambda = 0.07$ and $\lambda = 0, 2$ in Fig. 4.9 show that by applying a bandwidth saving of 35% as λ is changed from 0.07 to 0.2, not only the classification accuracy suffers little, but the visual quality of the reconstructed images also remains good.



(a) $\lambda = 0$ (b) $\lambda = 0.07$ (c) $\lambda = 0.2$

Figure 4.9: (a) Original image. 'Brownie' of size 64×64. (b) and (c) show reconstructed images after quantization and they look visually similar with SSIM [84] of 92.6% even though there is a 35% of bandwidth saving in (c) compared to (b).

4.5 CONCLUSIONS

In this chapter, we discussed methods to preserve higher accuracy in deep learning tasks while compressing data for bandwidth efficient transmission using conventional codecs such as JPEG2000. We proposed QuanNet as a novel end-to-end joint learning architecture for achieving this difficult constrained optimization problem. The QuanNet can be trained to optimize the quantization intervals of multiple sub-bands and multiple color channels to satisfy the bandwidth constraints without loss of accuracy. We proved the applicability of QuanNet by using deep learning task as image classification and observed significant rate-accuracy performance improvements.

Part II

LEARNING-BASED CODECS

5

END-TO-END IMAGE CLASSIFICATION AND COMPRESSION WITH VARIATIONAL AUTOENCODERS

So far, we discussed methods of efficiently using conventional image compression codecs such as JPEG2000 and proposed learning-based approaches to optimize the rate-accuracy performance of the codec for a given deep learning tasks. However, to overcome the infrastructural barrier of limited network bandwidth in cloud ML, existing solutions, including the proposed methods in chapters 3-4, have mainly relied on traditional compression that was historically engineered for human-end users instead of ML algorithms. Traditional codecs do not necessarily preserve features important to ML algorithms under limited bandwidth, leading to potentially inferior performance [31]. Hence, codecs must be end-to-end optimized for rate-accuracy performance based on a given task. While the non-differentiable design of traditional codecs constrain the end-to-end optimization, learning-based codecs can be end-to-end trained to minimize a given loss function. Chapters 5-6 explore such end-to-end optimization of learning-based codecs.

One way to design an $\mathcal{R}$-$\mathcal{C}$ optimal system for a given task is to end-to-end optimize system (codec and the task model) with task performed on the latent representation without relying on the reconstructed images [25, 85]. The other option is to jointly optimize system with task performed on reconstructed images [31]. In this case, the reconstructed images are not optimized for $\mathcal{R}$-$\mathcal{D}$. Hence a more meaningful way to extend this direction is to end-to-end optimized the system for rate-accuracy and distortion concurrently. Following the first avenue, this chapter investigates application-driven optimization of programmable commercial codec settings for networked learning tasks such as image classification. Based on the foundation of variational autoencoders (VAEs), we develop an end-to-end networked learning framework by jointly optimizing the codec and classifier without reconstructing images for given data rate (bandwidth). Compared with standard codecs including JPEG, JPEG2000 and BPG, the proposed VAE joint compression and classification framework achieves significant classification

accuracy improvements.. Following the second avenue, Chapter 6 explores leaning-based codec designs for improved rate-distortion-accuracy ($\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$) performance.

5.1 INTRODUCTION

The concept "Internet of Everything (IoE)" generalizes the idea of IoT to a broader paradigm. IoE nodes are connected by networks capable of communicating data generated from sensing and processing [86]. The power of IoE has already bolstered learning-intensive technologies such as self-driving cars, surveillance cameras, smart cities and smart transportation where DL applications are increasingly integrated with network services. As part of networked learning paradigm, these DL applications together with network services facilitate machine-to-machine and machine-to-edge/cloud server communications [87–91].

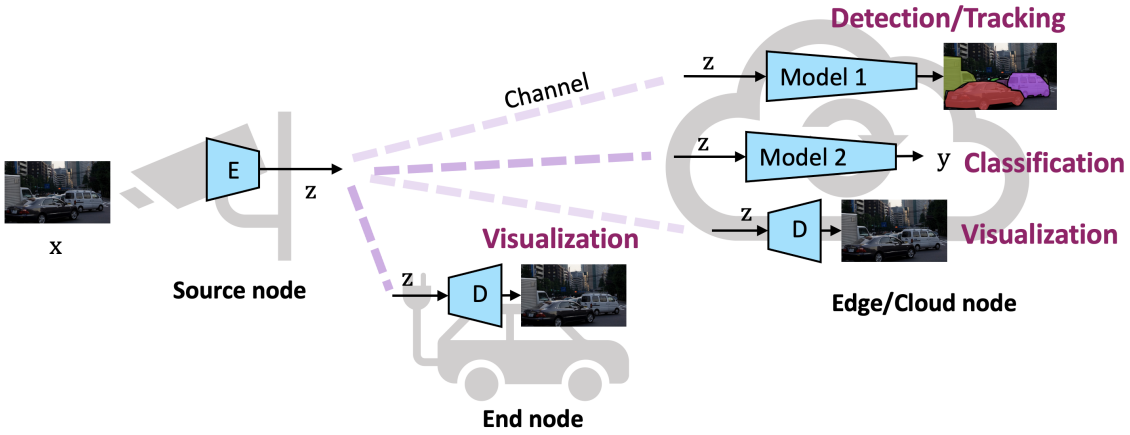


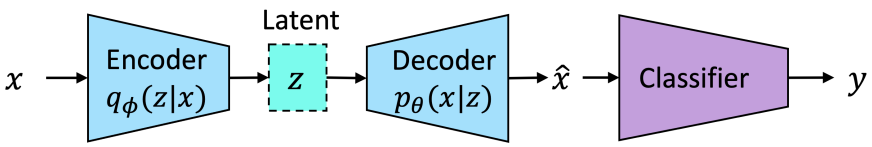
Figure 5.1: A networked ML instance: video/image frames from a camera are sent to a server for vehicle detection, traffic monitoring, and used by vehicles with an obstructed view for assisted/self-driving-related visualizations/decision-making. E: Encoder, D: Decoder.

One such example of networked learning is illustrated in Fig. 5.1. In automated inference systems such as pedestrian detection or object tracking for applications like self-driving, media data (image or video) collected by a source node is transported to network edge nodes and/or remote nodes to carry out more complex inference processes. The data transmission can be for multiple inference tasks (such as vehicle detection, traffic monitoring, or risk

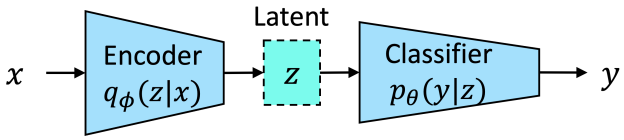
warning) at the edge servers or visualization purposes at cloud servers or end devices. For instance, video/image frames captured by a street camera are sent to a server for vehicular detection, crowd activity monitoring, etc., and used by smart vehicles with an obstructed view for assisted/self-driving-related visualizations/decision-making.

Given the explosive growth and deployment of IoT devices [39] and massive data collected by such devices [40], traditional cloud computing struggles to keep up with the demands of large IoT networks. Typical IoT configuration features source devices only as data collectors while cloud nodes are responsible for processing and analysis, hence, is limited by network link capacity, delay and losses, leading to long latency, unreliable inference and scalability issues [41–43]. Edge computing addresses these issues by deploying computing services in proximity to IoT devices [44]. Many image and video sensing devices such as smart phones, vehicular sensors and home/street cameras have sufficient resources to collect data and perform preliminary pre-processing and feature extraction, but not enough to implement full machine learning algorithms on-device [42]. With this new configuration of networked AI coupled with edge computing, such source devices can share a part of the computational burden by performing pre-processing and feature extraction, and deliver the extracted features to more powerful and resource rich edge/fog nodes for complex machine learning tasks.

When designing communication networks specifically targeting networked AI over edge/cloud for above such time-sensitive tasks, source data transmitted to the edge/cloud may fulfill two main requirements. On one hand, the data transport must achieve high coding efficiency ensuring low power usage and low latency. On the other hand, source data compression for transport must maintain high learning (task) accuracy. Since high learning accuracy demands high quality data with high coding rates [25, 31], there is a clear design trade-off between source coding rate and learning accuracy. This work designs a variational autoencoder (VAE)-based compression and learning framework to achieve a better coding rate and learning accuracy trade-off ensuring low power usage, low latency and low memory usage.



(a) VAE-based reconstruction, then classification



(b) Proposed: VAE-based classification

Figure 5.2: (a). VAE-based reconstruction before classification: Encoder transforms source image $\boldsymbol{x}$ to low dimensional $\boldsymbol{z}$. Decoder reconstructs image $\hat{\boldsymbol{x}}$ based on $\boldsymbol{z}$. (b). Proposed joint compression and classification: Without image reconstruction, classifier directly generates class label y based on $\boldsymbol{z}$.

Our focused investigation in this work is on an efficient image codec for classification applications. Conventional classification applications begin by reconstructing images of interest at the decoder before inference [92–94]. See Fig 5.2(a). We note the following two observations where the reconstruction can introduce inefficiency in cloud-based AI applications. First, when performing classification inference on the images compressed with image codecs that are originally optimized for rate-distortion performance targeting visualization applications, the classification accuracy suffers at high compression ratios under limited data rate [25, 30, 31, 85, 95]. Second, inference on images compressed with standard compression codecs such as JPEG2000 demonstrates non-negligible inference speed and accuracy degradation compared to with end-to-end-optimized joint classification and compression frameworks [30, 85] that bypass image reconstruction at decoders. Motivated by these observations, the proposed VAE-based classifier bypasses the reconstruction and is designed to naturally support end-to-end optimization for rate-accuracy performance. See Fig. 5.2(b).

In terms of device complexity, conventional video/image compression codecs such as VVC [32], HEVC [96] and JPEG [97] featuring a more complex encoder and a simpler decoder could be less desirable on low cost source devices. Instead, a more desirable data compression

and learning model for massive deployment in a network-AI edge/cloud should feature a simple encoder with low power and memory usage and a relatively more complex decoding and learning edge/cloud node. The trained encoder can reside in source devices with low computational power and memory to encode and compress data collections. Further, they can selectively compress and transmit task-important underlying features (e.g., latent) of the raw source data to other edge/cloud servers dedicated for decoding and learning ensuring $\mathcal{R}$-$\mathcal{C}$ performance.

The approaches in [25, 30, 85] are likely sub-optimal in terms of coding efficiency by optimizing only a part of pre-engineered JPEG2000 encoding-decoding pipeline. Addressing this, as part of end-to-end optimization of the codec, some recent DL-based image/video compression codecs showcase a learning-based encoder (autoencoder) that can be optimized to minimize a given loss function [31,34,98–101]. They demonstrated the importance of learning code word distribution for efficient compression instead of using fixed code word tables as in JPEG. However, the image encoders used in above works are still high in complexity, demanding high memory and power and resulting in lower inference speeds. Hence, they are not suitable for broad IoT deployment. Thus, we propose to use light-weight encoders and to model code word distributions for VAE-based classifier achieving high coding efficiency.

To achieve the aforementioned objectives in a networked AI environment, we propose a VAE-based joint compression and classification model (shown in Fig. 5.2(b)) that enables learning on the latent feature space to efficiently encode/compress and effectively classify images through end-to-end training. We aim to achieve high accuracy, fast inference, low bandwidth usage, and low latency over network links. Furthermore, our proposed framework features simple encoders while ensuring re-usability of the transmitted features. As main contributions of this work, we design a new information theoretical formulation of a VAE-based end-to-end, joint compression and classification framework. Towards this, we develop a power saving, low-complexity-encoder and a faster-inference-classifier for edge/cloud-based networked image classification applications.

We organize the rest of this chapter as follows. Sec. 5.2 reviews the related works on concept of autoencoder-based classifiers. Sec. 5.3 introduces our new proposal of a VAE-based end-to-end solution for classification by describing the basic learning model and the rate-accuracy function for joint classification-compression. In Sec. 5.4 we describe our experiment setup and present our test results. Additionally, Sec. 5.5 presents benchmark comparison of model complexity, inference speed, and power consumption for our proposed model. In Sec. 5.6, we discuss how to modify the proposed VAE-classifiers for simultaneous classification and reconstruction. See Appendix C for theoretical basis that connects the proposed VAE-based approached to frameworks such as information bottleneck [102, 103].

5.2 RELATED WORKS: VAE-BASED CLASSIFIER

When reviewing the VAE framework in Sec. 2.2, data reconstruction from $\mathbf{z}$ was typically the learning objective. We note, however, that for automated learning tasks such as image classification, reconstruction of $\mathbf{x}$ may not always be the end goal [31, 104] and may be unnecessary in many cases. For example, a practical security-related automated object detection system would rely on AI or deep learning algorithm trained on the latent vector $\mathbf{z}$ for object detection and/or recognition. Hence, the reconstruction of image frame $\hat{\mathbf{x}}$ is unnecessary for inference. Only at rare occasions such as evidence collection or inspection, the system may have to recover RGB frames for human visualization. Thus, the image reconstruction step can be performed only when required from the stored latent vectors.

Considering these practical AI/learning applications, we propose a VAE-based classifier by removing the VAE decoder in Fig. 5.2(a) to directly connect the latent code $\mathbf{z}$ with a data classifier to perform the learning task, as shown in Fig. 5.2(b). In this framework, variable y denotes the label of the output class based on $\mathbf{z}$. Our framework aims to jointly optimize the encoder and the classifier with end-to-end training.

Before presenting details of our proposed VAE framework, we first examine some related recent works on the joint learning of compression and classification based on autoencoders. Among others, one approach constructs a loss function by combining the latent entropy $H(\mathbf{z})$ with the classification cross entropy loss $\mathcal{L}_{\text{CE}}$ resulting from classification $\nu_0(\hat{\boldsymbol{x}})$ according to the reconstructed $\hat{\boldsymbol{x}}(\mathbf{z})$ from $\mathbf{z}$ (as shown in Fig. 5.2(a)).

$$\mathcal{L}(\boldsymbol{x}, y_{\text{gt}}) = \mathcal{L}_{\text{CE}}(y_{\text{gt}}, \nu_0(\hat{\boldsymbol{x}}(\mathbf{z})) + \beta H(\mathbf{z}|\mathbf{x}) \tag{5.1}$$

Here y_{gt} denoted the ground truth class of image sample $\boldsymbol{x}$. Clearly, this approach requires a reconstruction step prior to classification [105, 106].

Another approach is to construct the loss function by combining the latent entropy $H(\mathbf{z})$ with the classification CE loss based directly on the latent representation $\mathbf{z}$. Hence, the loss considering the classification $\nu_1(\mathbf{z})$ directly based on encoded $\mathbf{z}(\boldsymbol{x})$ is,

$$\mathcal{L}(\boldsymbol{x}, y_{\text{gt}}) = \mathcal{L}_{\text{CE}}(y_{\text{gt}}, \nu_1(\mathbf{z}(\boldsymbol{x}))) + \beta H(\mathbf{z}|\mathbf{x}). \tag{5.2}$$

Compared with the first approach, this formulation can potentially generate faster and more efficient training/inference models by skipping the reconstruction step [71]. For example, the authors of [25,30] introduced an image classification model by jointly training to optimize both the JPEG2000 encoder and the CNN classifier without image reconstruction. While achieving faster inference and higher accuracy, there is limited improvement since the proposed end-to-end framework only optimizes the quantization block of the encoder. Further, the majority of existing VAE-based image classification frameworks either still perform this unnecessary reconstruction step [92–94] or neglect the bandwidth aspect of the latent representation [107] making them impractical for wide IoT deployment in cloud-based image classification.

Table 5.1: Summary of notations

Notation	Meaning
$\boldsymbol{X}$	set of input images
$\mathbf{x}$	input image
$\hat{\mathbf{x}}$	reconstructed image
y_{gt}	ground truth label of image $\mathbf{x}$
y	estimated label of image $\mathbf{x}$
$\mathbf{z}$	latent vector representation of $\mathbf{x}$
$\hat{\mathbf{z}}$	quantized $\mathbf{z}$
d_ℓ	quantization level
$\mathcal{D}$	set of quantization levels d_ℓ
$p_{\mathbf{x},\boldsymbol{\theta}}$	probability distribution of $\mathbf{x}$ parameterized by $\boldsymbol{\theta}$
$p_{\mathbf{z}\|\mathbf{x},\boldsymbol{\theta}}$	conditional distribution of $\mathbf{z}$ parameterized by $\boldsymbol{\theta}$
$q_{\mathbf{z}\|\mathbf{x},\boldsymbol{\phi}}$	approximated distribution of $p_{\mathbf{z}\|\mathbf{x},\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\phi}$
$\hat{\rho}_{\boldsymbol{\theta}}(y\|\boldsymbol{x})$	estimated class label distribution with a model parameterized by $\boldsymbol{\theta}$ given the latent $\mathbf{x}$
$\eta_{\boldsymbol{\theta}}(y\|\boldsymbol{z})$	estimated class label distribution with a model parameterized by $\boldsymbol{\theta}$ given the latent $\mathbf{z}$
$P_{i,\ell_i}(\hat{\mathbf{z}}\|\boldsymbol{x})$	conditional probability of the latent value $\hat{z}_i$ that takes the value of the quantization level d_{ℓ_i}
$Q_{i,\ell_i}(y_{gt}\|\hat{\mathbf{z}})$	class label probability for each $\hat{z}_i$ that takes the value of d_{ℓ_i}
$\text{KL}(p,q)$	KL divergence between distributions p and q
$\text{CE}(y_{gt},y)$	cross entropy between distribution of y_{gt} and y
$H(p_{\mathbf{x},\boldsymbol{\theta}})$	entropy of variable $\mathbf{x}$ with distribution $p_{\mathbf{x},\boldsymbol{\theta}}$
β,γ	trade-off coefficients for rate, distortion.

5.3 A NEW VAE CLASSIFICATION FRAMEWORK

In this work, we focus on the second approach described in Eq. (5.2) and develop a novel VAE model for joint compression and classification encoding. Specifically, we design a VAE-based classifier for direct image classification based on the latent vectors without image reconstruction, as seen from Fig. 5.2(b). Targeting cloud-based classification over bandwidth-limited network data links, we implement adaptive entropy coders with context modeling aimed at improving overall $\mathcal{R}$-$\mathcal{C}$ performance trade-off. Further, we demonstrate that spatial

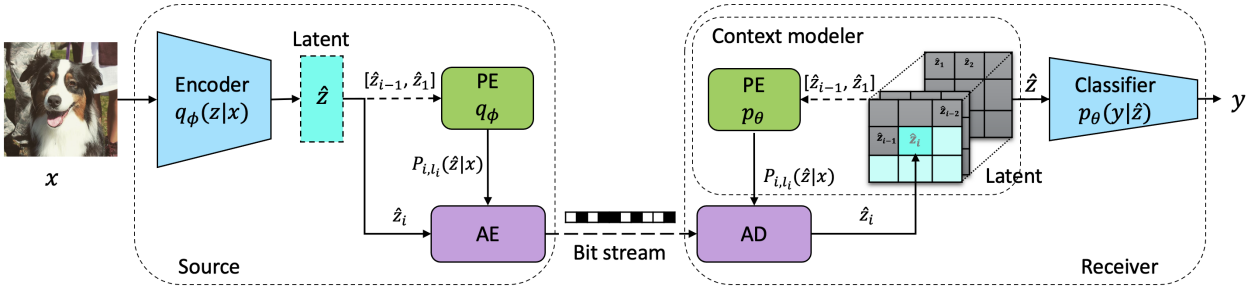


Figure 5.3: Overview of the proposed VAE classifier during inference: Quantized latent vector $\hat{\mathbf{z}}$ is encoded into bit stream by a context-adaptive arithmetic encoder (AE) assisted by probability estimator (PE). At receiver, probability of each symbol $\hat{z}_i$ (shown in cyan) is estimated by using a learned PE based on previously decoded latents $\hat{z}_{i-1}, \cdots, \hat{z}_1$ (shown in gray). Without groundtruth distribution of the latent elements $q_{\boldsymbol{\phi}}(\hat{\mathbf{z}}|\boldsymbol{x})$ at the receiver, PE learns to approximate $p_{\boldsymbol{\theta}} \approx q_{\boldsymbol{\phi}}$ during training.

domain RGB images can also be reconstructed for practical applications by fine-tuning a separate decoder under the proposed classification setting. In Table. 5.1, we summarize the notations used in the this chapter.

5.3.1 *Variational Bound and Loss for Classification*

The performance of a classifier can be measured by cross entropy loss from labeling. Consider an i.i.d. data sample $\boldsymbol{x}$ and corresponding label y that belongs to a set $\mathcal{Y}$ of unique labels, with $\{(\boldsymbol{x}, y)|\boldsymbol{x} \in \mathcal{D}, y \in \mathcal{Y}\}$. We can denote $\eta_{\boldsymbol{\theta}}(y|\mathbf{z})$ as the estimated label distribution with a model parameterized by $\boldsymbol{\theta}$ given the latent $\mathbf{z}$. For a latent $\mathbf{z}$ mapped from given $\boldsymbol{x}$, the estimated label distribution is simply

$$\hat{\rho}_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \int \eta_{\boldsymbol{\theta}}(y|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z} \tag{5.3}$$

Similar to 2.2, we introduce an arbitrary function $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ parameterized by $\boldsymbol{\phi}$.

$$\hat{\rho}_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \int \eta_{\boldsymbol{\theta}}(y|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z} = E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}} \left[\eta_{\boldsymbol{\theta}}(y|\mathbf{z}) \frac{p_{\boldsymbol{\theta}}(\mathbf{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x})} \right].$$

Taking $-\log$ on both sides leads to

$$-\log \hat{\rho}_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = -\log E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\left[\eta_{\boldsymbol{\theta}}(y|\mathbf{z})\frac{p_{\boldsymbol{\theta}}(\mathbf{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x})}\right]. \tag{5.4}$$

Applying Jensen's inequality, we write,

$$-\log \hat{\rho}_{\boldsymbol{\theta}}(y|\boldsymbol{x}) \leq -E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\log\left[\eta_{\boldsymbol{\theta}}(y|\mathbf{z})\frac{p_{\boldsymbol{\theta}}(\mathbf{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x})}\right] \leq -E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\log\eta_{\boldsymbol{\theta}}(y|\mathbf{z}) - E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x})}\right].$$

Following the definition of KL divergence, we write,

$$-\log \hat{\rho}_{\boldsymbol{\theta}}(y|\boldsymbol{x}) \leq -E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\log\eta_{\boldsymbol{\theta}}(y|\mathbf{z}) + \mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}). \tag{5.5}$$

We denote the ground truth class of sample $\boldsymbol{x}$ as y_{gt}. The true distribution ρ of y given $\boldsymbol{x}$ can be written as following by using the Kronecker delta function $\delta[\cdot]$

$$\rho(y|\boldsymbol{x}) = \delta[y - y_{\mathrm{gt}}] = \begin{cases} 1, & y = y_{\mathrm{gt}} \\ 0, & y \neq y_{\mathrm{gt}} \end{cases} \tag{5.6}$$

With the above notion, we write the cross entropy between the true and the estimated label distributions as

$$\mathrm{CE}_{y|\boldsymbol{x}}(\rho, \hat{\rho}_{\boldsymbol{\theta}}) = -\sum_{c\in\mathcal{Y}} \rho(y = c|\boldsymbol{x}) \log \hat{\rho}_{\boldsymbol{\theta}}(y = c|\boldsymbol{x}). \tag{5.7}$$

From (5.5) and (5.7), we can derive the variational bound for classification.

$$\begin{aligned}\mathrm{CE}_{y|\boldsymbol{x}}(\rho, \hat{\rho}_{\boldsymbol{\theta}}) &= -\sum_{c\in\mathcal{Y}} \rho(y = c|\boldsymbol{x}) \log \hat{\rho}_{\boldsymbol{\theta}}(y = c|\boldsymbol{x}) \\ &\leq -\sum_{c\in\mathcal{Y}} \rho(y = c|\boldsymbol{x})\{E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\log\eta_{\boldsymbol{\theta}}(y = c|\mathbf{z})\} + \sum_{c\in\mathcal{Y}} \rho(y = c|\boldsymbol{x})\{\mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}})\}\end{aligned}$$

The bound can be further simplified into

$$\mathrm{CE}_{y|\mathbf{x}}(\rho, \hat{\rho}_{\boldsymbol{\theta}}) \leq -\sum_{c \in \mathcal{Y}} \rho(y = c|\boldsymbol{x}) \{E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}} \log \eta_{\boldsymbol{\theta}}(y = c|\mathbf{z})\} + \mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}).$$

Using (5.6), we obtain the variational bound for classification:

$$\mathrm{CE}_{y|\boldsymbol{x}}(\rho, \hat{\rho}_{\boldsymbol{\theta}}) \leq -E_{q_{\mathbf{z}|\boldsymbol{x},\boldsymbol{\phi}}} \log \eta_{\boldsymbol{\theta}}(y_{\mathrm{gt}}|\mathbf{z}) + \mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}) \tag{5.8}$$

Following the approach [67] discussed in Sec. 2.2.1, we can define the β-VAE loss function for a single sample $\boldsymbol{x}$ as

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}, y_{\mathrm{gt}}) = E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}} \left[-\log \eta_{\boldsymbol{\theta}}(y_{\mathrm{gt}}|\mathbf{z})\right] + \beta E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}} \left[-\log p_{\boldsymbol{\theta}}(\mathbf{z}|\boldsymbol{x})\right]. \tag{5.9}$$

with a control parameter β. We can interpret the first term as the conditional classification loss and the second term as the coding cost or rate.

5.3.2 *Learning Model*

Our end-to-end compression-classification learning model consists of an encoder (E), a probability estimator (PE) and a classifier (CL) as shown in Fig. 5.3. Following the approach in [34], we use a context model to estimate the conditional probabilities in PE. Consider the following setting.

$$\mathbf{x} \xrightarrow[\boldsymbol{\phi}]{\mathrm{E}} \mathbf{z} \xrightarrow{\mathrm{Q}} \hat{\mathbf{z}} \xrightarrow[\boldsymbol{\theta}]{\mathrm{CL}} y \tag{5.10}$$

$\mathbf{x}$ is the input image to the model with dimensions $(w \times h \times 3)$. The encoder, parameterized by $\boldsymbol{\phi}$, maps $\mathbf{x}$ to the latent representation $\mathbf{z}$ of dimension $(\frac{w}{s} \times \frac{w}{s} \times K = m)$. A quantizer (Q) further maps $\mathbf{z}$ to $\hat{\mathbf{z}}$ by assigning each element z_i $(i = 1, \cdots, m)$ to the closest quantization

levels $\mathcal{D} = \{d_\ell,\ \ell = 1, \cdots,\ L\}$. For instance, multiple z_i values can be mapped to the same d_ℓ such that,

$$\hat{z}_i = Q(z_i) = \arg\min_{d \in \mathcal{D}} \|z_i - d\|_2. \tag{5.11}$$

The quantization centers $\{d_\ell\}$ are learned through training. The CL, parameterized by θ, takes the $\hat{\mathbf{z}}$ as the input and predicts the class label y for the given image sample $\boldsymbol{x}$. Taking quantization into account for practical applications, Eq. (5.9) can be rewritten as follows.

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}, y_{\text{gt}}) = E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\left[-\log \eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\hat{\mathbf{z}})\right] + \beta E_{q_{\hat{\mathbf{z}}|\mathbf{x},\boldsymbol{\phi}}}\left[-\log p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\boldsymbol{x})\right] \tag{5.12}$$

We now examine how to obtain each term for the loss function in Eq. (5.12).

5.3.3 *Rate Loss*

The second RHS term of Eq. (5.12) represents the cross entropy between two conditional distributions of $\mathbf{z}$: $p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\boldsymbol{x})$ and $q_{\boldsymbol{\phi}}(\hat{\mathbf{z}}|\boldsymbol{x})$. The distribution $q_{\boldsymbol{\phi}}(\hat{\mathbf{z}}|\boldsymbol{x})$ is deterministic and readily available after the deterministic encoding step (E). During training, we estimate the distribution $p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\mathbf{x})$ with a PE using a conditional context model following [34, 67]. The latent representation $\hat{\mathbf{z}}$ as discussed in Sec. 5.3.2 is a 3D tensor containing m number of latent elements. We can index this 3D tensor $\hat{\mathbf{z}}$ given $\boldsymbol{x}$ as a vector in raster-scan order and write $p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\boldsymbol{x})$ as

$$p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\boldsymbol{x}) = \prod_{i=1}^{m} p_{\boldsymbol{\theta}}(\hat{z}_i|\hat{z}_{i-1}, \cdots, \hat{z}_1, \boldsymbol{x}). \tag{5.13}$$

With PE, we can efficiently estimate the conditional probability of the latent value $\hat{z}_i$ that takes the value of the quantization center d_{ℓ_i} which can be defined as $P_{i,\ell_i}(\hat{\mathbf{z}}|\boldsymbol{x}) = p_{\boldsymbol{\theta}}(\hat{z}_i =$

$d_{\ell_i}|\hat{z}_{i-1}, \cdots, \hat{z}_1, \boldsymbol{x})$ for notional simplicity. With this notation, can write the approximate coding rate ($\mathcal{L}_{\mathrm{R}}$) as follows:

$$\mathcal{L}_{\mathrm{R}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}) = \mathrm{CE}_{\hat{\mathbf{z}}|\boldsymbol{x}}(q_{\boldsymbol{\phi}}, p_{\theta}) = E_{q_{\hat{\mathbf{z}}|\mathbf{x},\boldsymbol{\phi}}}\left[-\log p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\boldsymbol{x})\right] = -\sum_{i=1}^{m}\sum_{\ell_i=1}^{L} q_{\boldsymbol{\phi}}(\hat{z}_{i,\ell_i}|\boldsymbol{x}) \log P_{i,\ell_i}(\hat{\mathbf{z}}|\boldsymbol{x}). \tag{5.14}$$

Here $q_{\boldsymbol{\phi}}(\hat{z}_{i,\ell_i}|\boldsymbol{x})$ is the probability of $\Pr(\hat{z}_i = d_{\ell_i}|\boldsymbol{x})$. Any $\hat{z}_i$ can take only one particular quantization value d_{ℓ_i}.

Similar to [34], through end-to-end training, we learn the distribution $p_{\theta}(\hat{\mathbf{z}}|\boldsymbol{x})$ in order to optimally approximate $p_{\boldsymbol{\theta}} \approx q_{\boldsymbol{\phi}}$. Through context-based adaptive arithmetic encoding process (CABAC) [108], $q_{\boldsymbol{\phi}}(\hat{\mathbf{z}}|\boldsymbol{x})$ is available for the probability estimation, only at the encoder. Once the encoded bit-stream is received at the context-based adaptive arithmetic decoder, the learned conditional probability $p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\boldsymbol{x})$ is used for probability estimation to recover $\hat{\mathbf{z}}$ whereas $q_{\boldsymbol{\phi}}(\hat{\mathbf{z}}|\boldsymbol{x})$ is not available [109]. For this reason, we learn the distribution $p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\boldsymbol{x})$ to approximate $p_{\boldsymbol{\theta}} \approx q_{\boldsymbol{\phi}}$ which makes $p_{\boldsymbol{\theta}}$ the "context modeler" as commonly referred in the literature [108, 109]. Refer to Fig. 5.3. When the learning succeeds by training, with Eq. (5.14), we observe that the coding rate $\mathcal{L}_{\mathrm{R}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x})$ reduces to the entropy $H(\hat{\mathbf{z}}|\boldsymbol{x})$.

5.3.4 *Classification Loss*

We interpret the first RHS term of Eq. (5.12) as the classification loss $\mathcal{L}_{\mathrm{CL}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}, y_{gt})$. Following the rate approach as given in Sec. 5.3.3, we can rewrite the classification loss term as follows.

$$\mathcal{L}_{\mathrm{CL}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}, y_{\mathrm{gt}}) = E_{q_{\hat{\mathbf{z}}|\mathbf{x},\boldsymbol{\phi}}}\left[-\log \eta_{\boldsymbol{\theta}}(y_{\mathrm{gt}}|\hat{\mathbf{z}})\right] = -\sum_{i=1}^{m}\sum_{\ell_i=1}^{L} q_{\boldsymbol{\phi}}(\hat{z}_{i,\ell_i}|\boldsymbol{x}) \log Q_{i,\ell_i}(y_{\mathrm{gt}}|\hat{\mathbf{z}})$$

Here we have to estimate the class label probability for each $\hat{z}_{i,\ell_i}$ which is defined as $Q_{i,\ell_i}(y_{\text{gt}}|\hat{\mathbf{z}}) = \eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\hat{z}_i = d_{\ell_i})$. We note that estimating the conditional label probability given each individual $\hat{z}_i$ can be challenging. On the other hand, given $\hat{\mathbf{z}}$, we can efficiently predict the class label using a trained CNN classifier such as ResNet [15]. Hence, we can easily estimate $\log \eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\hat{\mathbf{z}})$ given the entire sequence of $\hat{\mathbf{z}}$ rather than for one individual element $\hat{z}_i$ therein. In standard stochastic gradient optimizers, we have only one sequence sample $\hat{\mathbf{z}}$ at a time. Hence, the stochastic expectation of $E_{q_{\hat{\mathbf{z}}|\mathbf{x},\boldsymbol{\phi}}}[-\log \eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\hat{\mathbf{z}})]$ can be approximated simply by $-\log \boldsymbol{\eta}_{\theta}(y_{\text{gt}}|\hat{\mathbf{z}})$ [66]. We can reduce this approximation error by iterating sufficiently over same $\boldsymbol{x}$. This simple approximation leads to general classification loss,

$$\mathcal{L}_{\text{CL}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}, y_{\text{gt}}) = E_{q_{\hat{\mathbf{z}}|\mathbf{x},\boldsymbol{\phi}}}[-\log \eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\hat{\mathbf{z}})] \approx -\log \eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\hat{\mathbf{z}}). \tag{5.15}$$

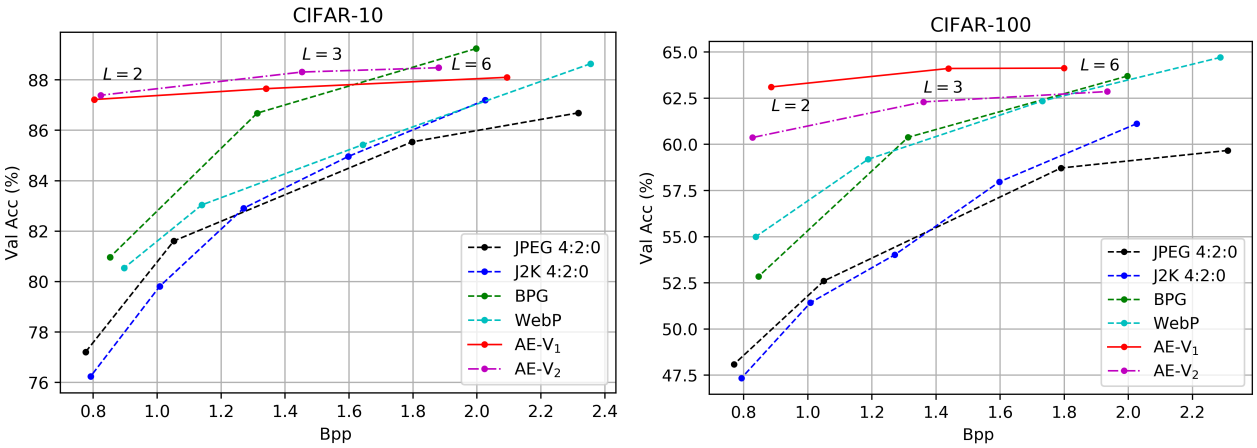
With this formulation, we express the total loss $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}, y_{\text{gt}})$ of the joint classification and compression model for image sample $\boldsymbol{x}$ as,

$$\begin{aligned}\mathcal{L}_{\theta,\boldsymbol{\phi}}(\boldsymbol{x}, y_{\text{gt}}) &= \mathcal{L}_{\text{CL}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}, y_{\text{gt}}) + \beta \mathcal{L}_{\text{R}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}) \\ &\approx -\log \eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\hat{\mathbf{z}}) - \beta \sum_{i=1}^{m} \sum_{\ell_i=1}^{L} q_{\boldsymbol{\phi}}(\hat{z}_{i,\ell_i}|\boldsymbol{x}) \log P_{i,\ell_i}(\hat{\mathbf{z}}|\boldsymbol{x})\end{aligned} \tag{5.16}$$

Similar to the rate-distortion loss in reconstruction [34, 67, 110], β controls the rate versus mis-classification trade-off.

5.4 EXPERIMENTS AND RESULTS

This section presents several experiment results of classification using the proposed VAE approach of Sec. 5.3. We tested the accuracy of classification on three well-known data sets: CIFAR-10 [111], CIFAR-100 [112] and ImageNet-1k [77]. CIFAR-10 consists of 10 classes



(a) CIFAR-10 (b) CIFAR-100

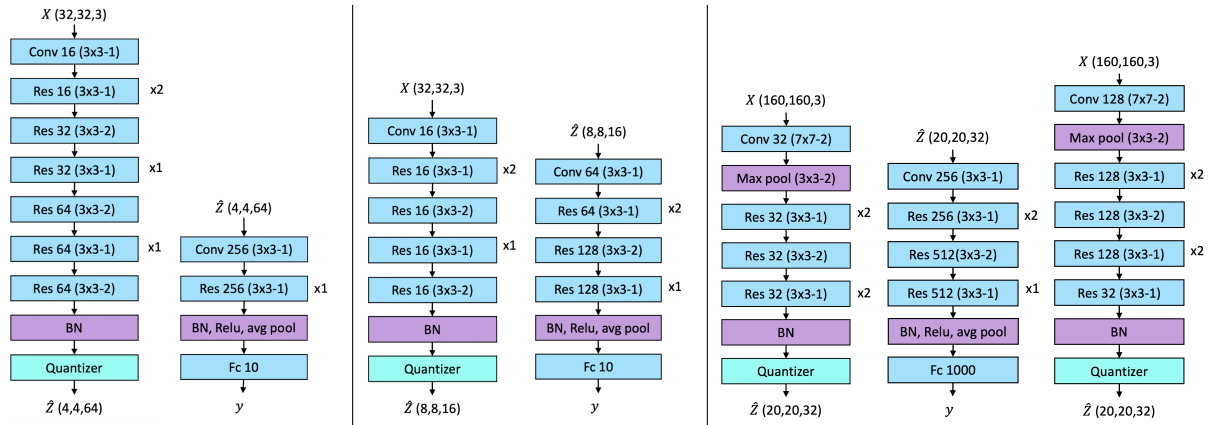
Figure 5.4: Classification accuracy vs rate results for end-to-end compression and classification on (a) CIFAR-10 and (b) CIFAR-100 data sets. The proposed VAE-based classification framework outperforms popular commercial image compression codecs in terms of rate-accuracy, at lower bandwidths.

with 5000 images for training and 1000 images for testing per class. CIFAR-10 images are in RGB format of size 32×32. Similar to CIFAR-10, CIFAR-100 data set contains 50k training and 10k test RGB images of size 32×32 in 100 classes. ImageNet-1k consists of 1000 classes, each of which consists of up to 1300 training images and 50 validation images of mostly 256×256 RGB.

Our experiments used the same quantizer and the context model as [34] to estimate probabilities. Instead of separately training the encoder-decoder and the context model (PE) as in [34], the training of PE takes place simultaneously with encoder and classifier training.

5.4.1 *VAE-based Joint Compression and Classification models*

When designing the joint model, we began with a ResNet and split into 2 parts as Encoder (E) and classifier (CL). Next, we added a batch normalization layer and a quantizer block to the encoder and further modified the number of filters and strides. For each CIFAR-10, CIFAR-100 and ImageNet-1k data sets, we considered 2 different E-CL combinations. Figs. 5.5(a)-(g) depict the structural details of 4 proposed VAE models labeled as AE-V_1,



(a) AE-V_1 Encoder (b) AE-V_1 Classifier (c) AE-V_2 Encoder (d) AE-V_2 Classifier (e) AE-V_3 Encoder (f) AE-V_3/V_4 Classifier (g) AE-V_4 Encoder

Figure 5.5: Encoders and classifiers for the proposed end-to-end compression and classification models AE-V_1, AE-V_2, AE-V_3 and AE-V_4. "Conv 16 (3x3-2)" represents a 2D convolution block with 32 filters of size 3×3 and stride of 2. "Res 16 (3x3-2)" represents a basic ResNet block [15] with down-sampling factor 2.

AE-V_2, AE-V_3 and AE-V_4. We use typical notations in DL to denote their components. For example, "Conv 16 (3×3-2)" represents a 2D convolution block with 32 filters of size 3×3 and a stride of 2. "Res 16 (3×3-2)" denotes a basic ResNet block [15] with a down-sampling factor of 2.

To compare the performance gains of the proposed approach, we mainly used JPEG (4:2:0)[1] as the benchmark technique to compress images within the data sets at different quality (Q) values. One can vary the quality value $Q \in [1, 100]$ during compression to set image quality. We then trained a ResNet classifier based on these compressed images for each Q value. To calculate the required channel bandwidth (i.e., image size), we averaged the bits-per-pixel (bpp) for each coded image over the data set. For fair comparison, we only consider the data bits of the images according to bpp without counting packet headers. Following similar steps, we generated rate-accuracy performance for popular standard codecs: JPEG2000[2], WebP[3] and BPG[4], to be used as baselines.

[1] http://www.openjpeg.org/
[2] https://kakadusoftware.com/
[3] https://developers.google.com/speed/webp/download
[4] https://bellard.org/bpg/

In the joint compression and classification model, we vary the number of quantization centers (L) to compress images at different bpp values without changing the E and CL model architectures. As proposed in [34], one can alter the encoder-decoder/classifier architecture to generate latent of different dimensions to obtain the requisite bpp as well.

We selected $\beta = 2$ for all experiments. During model training, we used a 'Momentum' Optimizer for the E-CL part of the network and an 'Adam' optimizer for the PE with a learning rate of 0.0005 and reduced the learning rate by $\times 0.1$ at 0.25 and 0.75 of total epochs over 90 epochs. In contrast to [34], we concurrently train both E-CL and PE to also enable back-propagation from PE to E. This configuration facilitates model convergence at lower bpp.

5.4.2 *CIFAR-10 Experiments*

For CIFAR-10, we used the models AE-V_1 and AE-V_2 of different complexities. Figs. 5.5(a)-(d) provide the structural details of both VAE models. As the baseline classification model, we used a ResNet-18. Fig. 5.4(a) shows the result for CIFAR-10 data set with different combinations of E-CL in comparison with popular standard codecs: JPEG, JPEG2000, WebP and BPG. Both AE-V_1 and AE-V_2 achieve substantial performance improvement over the traditional JPEG (4:2:0) in terms of rate (bpp) and accuracy trade-off. At 0.8 bits/pixel, our joint compression-classification accuracy improves from 77% to 87%. More importantly, this performance improvement is achieved with significantly lighter-weight models, resulting impressive inference speed improvements and power savings compared to the baseline classifiers as will be discussed in Sec. 5.5. Further, compared to JPEG2000, WebP and BPG, the proposed VAE-based codecs maintain the $\mathcal{R}$-$\mathcal{C}$ performance specially at lower bandwidth end. In particular, both VAE-based models demonstrate over 5% accuracy improvement at 0.9 bpp compared to BPG.

5.4.3 *CIFAR-100 Experiments*

As the baseline classification model, we used a ResNet-18 model with a fully connected layer of 100 neurons instead of 10 in the original model designed for CIFAR-10. For the VAE models for classification, we used AE-V_1 and AE-V_2, each with same Encoder (shown in Figs. 5.5(a), (c)) and the Classifier (Figs. 5.5(b), (d)) with a fully connected layer of 100 neurons (Fc 100) instead of Fc 10. For training, we follow the same settings as in CIFAR-10. Fig. 5.4(b) compared the rate-accuracy performance of AE-V_1 and AE-V_2 models for CIFAR-100 validation set. The proposed method achieves similar improvement as observed in CIFAR-10.

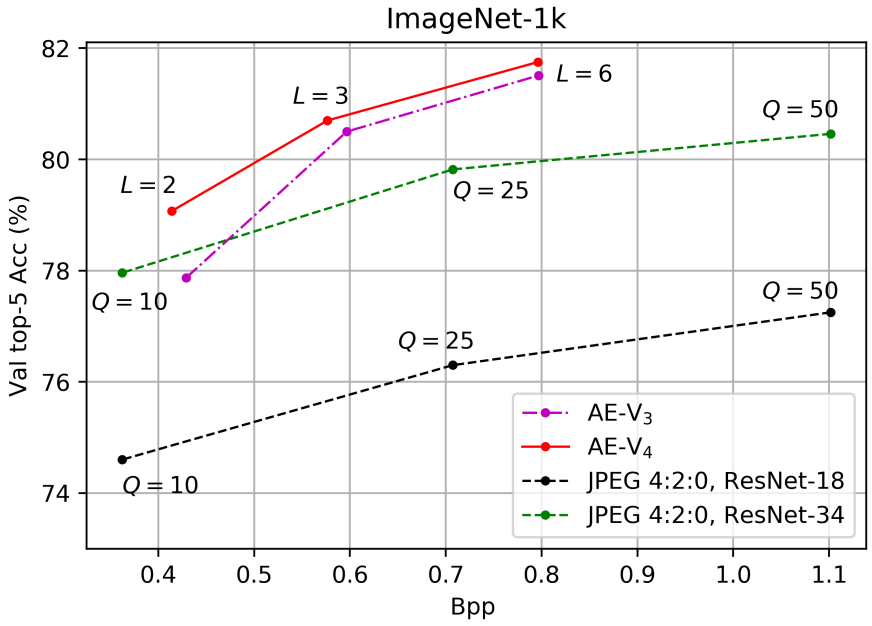


Figure 5.6: Classification accuracy vs rate on ImageNet-1k for end-to-end compression and classification. The proposed VAE based compression and classification framework (AE-V_4) significantly outperforms JPEG commercial image compression codecs in terms of rate-accuracy.

5.4.4 *ImageNet Experiments*

For ImageNet experiments, we followed the same process as used for CIFAR-10 to generate the JPEG (4:2:0) baseline. As a part of data augmentation, we resized the images to 256×256 and cropped them down to 160×160 in training and testing to obtain the desired dimensions of the latent $\hat{\mathbf{z}}$. We propose modified VAE models AE-V_3 and AE-V_4 for ImageNet. See Figs. 5.5(e)-(g) for model architectures.

For fairness in performance comparison, we fine-tuned the baseline ResNet models with the JPEG encoded images at each quality (Q) to record the best accuracy achievable at given quality levels when obtaining baseline performance results. For the same reason, we do not include packet header in bpp calculation.

Testing the **E-CL** combinations AE-V_3 and AE-V_4 on ImageNet generates the results of Fig. 5.6. The performance of the two shallower VAEs is compared against the benchmark ResNet-18 and ResNet-34 operated on JPEG encoding. Our results demonstrate clear performance improvement in terms of rate-accuracy trade-off, without increasing complexity as discussed in Sec. 5.5. In fact, the proposed VAE delivers comparable performance as conventional ResNet-34 that requires 21.8M parameters which is twice as many compared to AE-V_3 and AE-V_4 models.

5.5 COMPLEXITY COMPARISON

Simpler encoder models are practically more favorable for wide deployment on low cost devices, which are severely constrained in memory, power, and computation capacity. In this section, we compare the complexity of the proposed models in terms of model size (number of parameters), inference speed and power consumption to the baseline models.

5.5.1 *Model size comparison*

Considering practical constraints in low cost sensors, our encoder design has significantly fewer number of parameters in comparison with the typical cloud-based classifier. See Table 5.2 for a complexity comparison in terms of number of parameters for the VAE models used in CIFAR-10 and ImageNet-1k experiments. Percentages provided in the table indicates the reduction of the number of parameters compared to the baseline classifier.

Table 5.2: Complexity comparison of the models for CIFAR-10 and ImageNet. All parameters are in millions (M). The percentage reduction is calculated compared to the baseline.

Model	Encoder	Decoder	Classifier	Baseline
CIFAR-10:				
AE-V_1	0.245 (**65**%↓)	–	0.372 (**47**%↓)	0.704 ResNet-18
AE-V_2	0.025 (**96**%↓)	–	0.687 (**2**%↓)	
ImageNet-1k:				
AE-V_3	0.100 (**99**%↓)	–	11.35 (**3**%↓)	11.7 ResNet-18
AE-V_4	1.550 (**87**%↓)	–	11.35 (**3**%↓)	
Theis-2017 [83]	1.403 (**88**%↓)	1.630 (**86**%↓)	11.7	
Factorized priors [113]	3.506 (**70**%↓)	3.505 (**70**%↓)	11.7	
Scale hyper-priors [100]	5.903 (**50**%↓)	5.902 (**50**%↓)	11.7	
ImageNet-1k:				
AE-V_3	0.100 (**99**%↓)	–	11.35 (**48**%↓)	21.8 ResNet-34
AE-V_4	1.550 (**99**%↓)	–	11.35 (**48**%↓)	

All the proposed encoder and classifier models show parameters savings compared to their corresponding baseline models. The achieved parameter savings range from 65%−99% for encoders and 2%−48% for classifiers. Such balance of complexity reductions are highly practical for wide IoT deployment since classifiers at the server side are equipped with more resources compared to the source encoders at the embedded devices.

Further in Table. 5.2 we compare the model size of the proposed VAE-based classifiers to popular image compression codecs: Scale hyper-priors [100] and Factorized priors [113]. Our

proposed encoder models of VAE-based image classifiers are significantly simple compared to encoders of learning-based image compression codecs. For instance, AE-V_3 encoder is only 1.7% compared to Scale hyper-priors encoder. This is highly favorable for wide deployment of embedded devices that are limited in on-device memory. On the other hand, since our proposed VAE-based models do not perform image reconstruction, no of parameters of the models residing at the server is also highly reduced.

Task vs Model size

When comparing the model complexity versus task difficulty, more challenging classification tasks often require more complex encoders. Note that both models AE-V_3 and AE-V_4 use the same network architecture as the classifier for ImageNet data set. Furthermore, from Table 5.2, it is clear that the proposed VAE classifiers of AE-V_3 and AE-V_4 have the same complexity in terms of the number of parameters while AE-V_4 model uses a relatively more complex encoder than AE-V_3. Since complex encoders with more parameters are capable of extracting more discriminate features, Fig. 5.6 shows AE-V_4 with higher rate-classification performance than AE-V_3, especially when handling more challenging tasks such as ImageNet classification with 1000 classes.

We observe similar trend in rate-classification in Fig. 5.4(b) for CIFAR-100 data set with 100 classes: A relatively increase of encoder complexity may achieve better performance than a simple encoder through feature extraction for classification. Note in Table 5.2 that the model complexity in number of parameters is $\times 10$ smaller for AE-V_2 when compared to AE-V_1. However, for relatively simple classification task such as CIFAR-10 with only 10 classes, a simpler encoder used in AE-V_2 is stronger to preserve enough features to generate the latent vector. In fact, even in such case, having a relatively complex classifier with more parameters to process the extracted features has some benefit, although the resulting rate-classification performance improvement is rather modest.

5.5.2 *Inference speed comparison*

Table 5.3: Speed comparison in terms of "ips". The inference results are based on a NVIDIA Titan-V GPU. The proposed VAE-based classifier is ×1.5 and ×2.25 faster compared to ResNet-18 and ResNet-34 baselines.

Model	Encode (**E**)	Classifier (**CL**)	**E-CL**	Baseline **CL**
AE-V_1	1702	20283 (×**13.1**↑)	1570(↑)	1547 ResNet-18
AE-V_2	2114	6524 (×**4.2**↑)	1596(↑)	
AE-V_3	4691	5122 (×**1.5**↑)	2448(↓)	3330 ResNet-18
AE-V_4	2417		1642(↓)	
AE-V_3	4691	5122 (×**2.25**↑)	2448(↑)	2268 ResNet-34
AE-V_4	2417		1642(↓)	

We also observe considerable reduction in inference time by directly classifying on the latent/feature maps without reconstruction. In Table 5.3 we list the inference speed in terms of average images per second (ips) for CIFAR-10 test set and ImageNet-1k validation set.

The proposed end-to-end-trained AE-V_1 and AE-V_2 classifiers record ×13.1 and ×4.2 speed gains on CIFAR-10 data set, respectively, compared to the ResNet-18 classifier. Image patches used at inference have the size of 32×32. Similarly, our proposed classifier is ×1.5 faster in comparison with the ResNet-18 baseline and ×2.25 faster in comparison with ResNet-34 baseline, on ImageNet-1k data set where image patches have the size of 160×160. Since the inference speed of the encoder for the baseline can vary based on the codec (JPEG, BPG etc.), we only highlight the inference speed gains of the classifiers (**CL**) for fairness. Moreover, due to the simplicity of the proposed AE-V_1 and AE-V_2 encoders, the inference speed including the encoding time (**E-CL** column) surpasses the baseline CL speed calculated even without the encoding time.

5.5.3 *Power savings comparison*

When measured on the same device (GPU), the number of floating point operations per second (FLOPS) is directly proportional to the power consumption. The proposed encoders are not only less complex in terms of number of parameters, but also demand significantly lower number of arithmetic operations which is an essential feature for IoT, mobile and wireless applications [114]. In Table 5.4, we list the required computational operations in floating point operations per second (FLOPs) for the proposed encoders. We indicate the power savings as a percentage compared to the baseline.

Table 5.4: Number of FLOPs ($\times 10^9$) comparison for the proposed models.

Model	Encoder	Decoder	Classifier	Baseline
Input: 32×32×3				
AE-V_1	0.037 (**93**%↓)	–	0.005 (**99**%↓)	0.557
AE-V_2	0.013 (**97**%↓)	–	0.013 (**97**%↓)	ResNet-18
Input: 160×160×3				
AE-V_3	0.107 (**88**%↓)	–	1.229 (**32**%↑)	
AE-V_4	1.399 (**50**%↑)	–	1.229 (**32**%↑)	0.928
Theis-2017 [83]	2.102 (**127**%↑)	2.551 (**175**%↑)		ResNet-18
Factorized priors [113]	6.140 (**561**%↑)	6.140 (**561**%↑)	0.928	
Scale hyper-priors [100]	6.355 (**584**%↑)	6.355 (**584**%↑)	0.928	
Input: 160×160×3				
AE-V_3	0.107 (**94**%↓)	–	1.229 (**34**%↓)	1.873
AE-V_4	1.399 (**25**%↓)	–	1.229 (**34**%↓)	ResNet-34

The proposed encoders and classifiers of AE-V_1 and AE-V_2 models achieve power savings in the range of 93%−97% compared to ResNet-18 classifier. AE-V_3 and AE-V_4 models with larger input image sizes show less power savings in the classifier compared to the encoder due to the high number of parameters. However, this low power savings are compensated by the resource rich servers at the cloud. More importantly, the proposed encoders display significant power savings that enable wide deployment of power constrained low-end source embedded devices. Modern smart phones are capable of providing over 10×10^9 FLOPs [114, 115].

Similar to model size, our proposed VAE-based classifier models demands significantly less computational power compared to other learning-based compression codecs as shown in Table 5.4. For instance, AE-V_3 and AE-V_4 encoders consumes only 1.7% and 22% respectively compared to Scale hyper-priors encoder.

5.5.4 *Comparison to Torfason-2018 [78]*

In this section, we further compare the proposed VAE-based classifiers against Torfason-2018 [78] in terms of no of FLOPs required during inference for image classification task. The work [78] proposed to remove the expensive decoders during deep learning tasks such as image classification and segmentation and showed that same rate-accuracy performance can be achieved without image reconstruction. They further showed that for image classification task, initial convolution and ResNet layers can be removed from traditional ResNet [15] classifiers. By introducing such changes, the authors achieved $\times 1.5 - \times 2$ FLOP reductions compared to baseline encoder-decoder-classifier work flow.

During the experiments, the authors of [78], used the same encoder architecture in [83] without any modification and obtained FLOP reductions by skipping image decoding and with complexity reduced classifiers. We note two main benefits of our proposed method over [78]. First, to support the cloud-based image classification setting with low-resource embedded sources, the encoders of the proposed VAE-based models are significantly low in power demand compared to those of [78] and [83] as shown in Table 5.4. Second, when comparing the total FLOP requirement for both encoder and the classifier, our proposed VAE-based models show further notable power savings compared to [78]. Furthermore, since the classifiers are generally located at resource-rich server nodes during cloud-based inference, we argue that complexity reduction of the classifier sacrificing rate-accuracy performance is less meaningful.

In Table 5.5, we compare the FLOPs required by the encoder, decoder and the classifier of the considered models for image sizes 160×160 and 224×224. Compared to AE-V_3 model, the encoder-classifier model proposed in [78] demands ×2.68 and ×1.89 more FLOPs for image sizes 160×160 and 224×224 respectively.

Table 5.5: Number of FLOPs ($\times 10^9$) comparison for the proposed models. Power savings compared to the encoder of [78] are given as a percentage. In last column, the increase of FLOPs are calculated compared to the total FLOPS required by AE-V_3.

Model	Encoder	Decoder	Classifier	Total
Input: 160×160×3				
AE-V_3	0.107 (**95**%↓)	–	1.229 (**42**%↓)	1.336
AE-V_4	1.399 (**33**%↓)	–	1.229 (**42**%↓)	2.628 (× 1.97)
Theis-2017 [83]	2.102	2.551 (↑)	0.928 (Res-18)	5.581 (× 4.18)
Theis-2017 [83]	2.102	2.551 (↑)	1.873 (Res-34)	6.526 (× 4.88)
Torfason-2018 [78]	2.102	–	1.475 (cRes-39)	3.577 (× 2.68)
Input: 224×224×3				
AE-V_3	0.215 (**95**%↓)	–	3.453 (**14**%↓)	3.668
AE-V_4	2.746 (**31**%↓)	–	3.453 (**14**%↓)	6.199 (× 1.69)
Theis-2017 [83]	4.003	4.744 (↑)	1.817 (Res-18)	10.56 (× 2.88)
Theis-2017 [83]	4.003	4.744 (↑)	3.667 (Res-34)	12.41 (× 3.38)
Torfason-2018 [78]	4.003	–	2.95 (cRes-39)	6.953 (× 1.89)

5.6 DISCUSSION: JOINT COMPRESSION AND CLASSIFICATION WITH SIMULTANEOUS RECONSTRUCTION

Since encoded images may serve the dual-use of human visual perception and machine learning in certain practical applications, we investigate the feasibility of adapting the VAE-based classifier as a dual-use codec for efficiently encoding image features delivered to the cloud for both accurate image classification and image reconstruction of sufficiently high quality (PSNR).

Thus far, our proposed joint compression and classification model allows the classifier to operate directly in latent space. We bypass the image reconstruction step unlike the

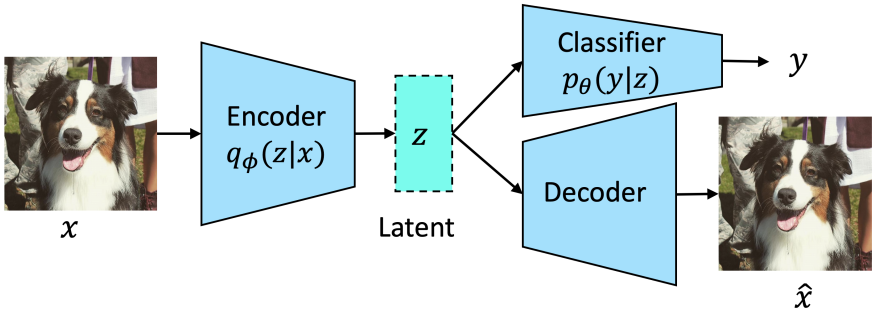


Figure 5.7: Proposed joint classification and compression model with reconstruction. Encoder transforms input $\boldsymbol{x}$ to a latent vector $\boldsymbol{z}$ that is optimized for rate-classification-distortion performance.

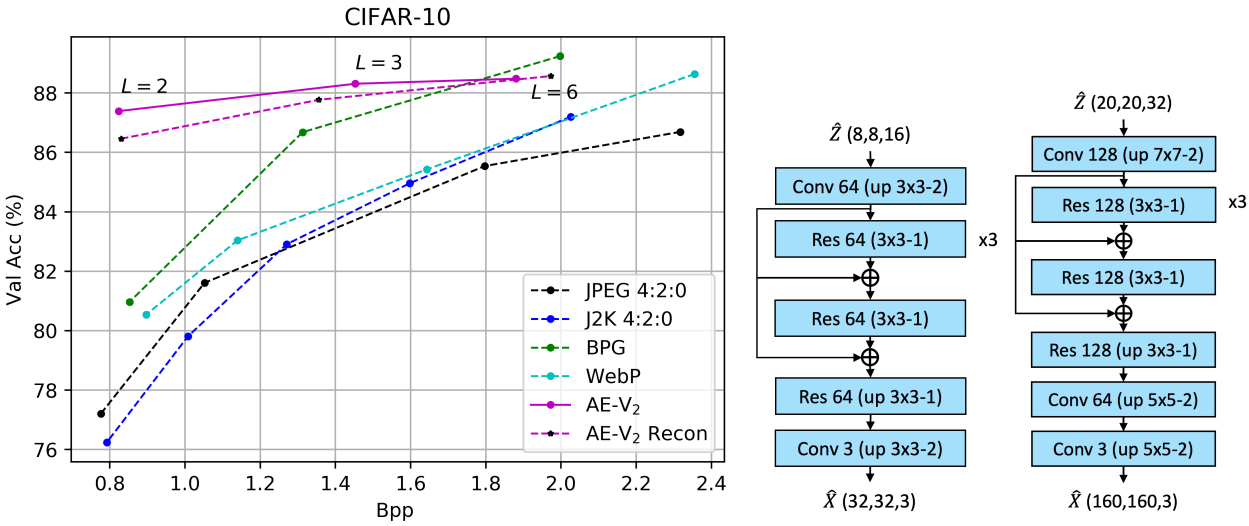
image/video systems [105] that perform classification on reconstructed data. In practical systems, however, there also exist several application scenarios which may require both autonomous image classification and image reconstruction for users. For example, RGB images may need to be stored on the cloud to be retrieved later after successful classification. These reconstructed and stored RGB images may be required for other subsequent learning tasks such as object detection and segmentation trained on RGB images. To accommodate such potential dual compression objectives of simultaneous classification and reconstruction, we further modify the current VAE to incorporate a parallel decoder at the remote node (or cloud).

As shown in Fig. 5.7, we include a basic DL-based image decoder (D) to the proposed joint compression and classification model. This framework of joint compression and classification model with reconstruction (E-CL-D) can be trained similarly to the proposed VAE. In consideration of the reconstruction accuracy, we construct a modified loss function of

$$\mathcal{L} = \mathcal{L}_{\mathrm{CL}} + \beta\mathcal{L}_{\mathrm{R}} + \gamma\mathcal{L}_{\mathrm{D}}. \tag{5.17}$$

We use MSE between $\mathbf{x}$ and $\hat{\mathbf{x}}$ as the distortion loss $\mathcal{L}_{\mathrm{D}}$ and empirically choose a control parameter γ smaller than β to favor high classification accuracy over rate-distortion.

We fine-tuned the proposed joint architecture in Fig. 5.7 with the above modified loss on CIFAR-10 data set. Fig. 5.8(a) shows the classification accuracy for E-CL-D model for



(a) Rate-accuracy (b) AE-V$_2$ Decoder (c) AE-V$_3$ Decoder

Figure 5.8: (a) Classification accuracy vs rate on CIFAR-10 for joint compression-classification with reconstruction: AE-V_2-Recon. Note that rate-accuracy performance of AE-V_2-Recon is still sufficiently good compared to popular commercial codecs, at lower rates. (b)-(c) Decoder design for AE-V$_2$ and AE-V$_3$. 'up' indicates up-sampling with transposed convolutions.

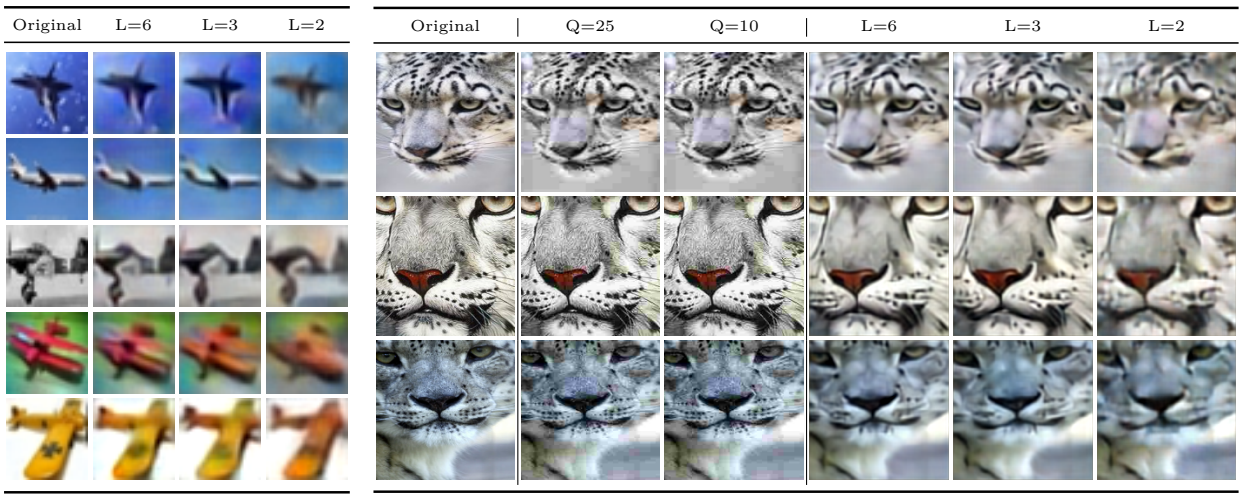
CIFAR-10. Note that the classification accuracy loss due to reconstruction is smaller compared to the $\mathcal{R}$-$\mathcal{C}$ gain from standard JPEG, JPEG2000, WebP and BPG codecs. We repeated the joint classification and reconstruction experiments on ImageNet-1k with a slight modification to the training strategy to further preserve $\mathcal{R}$-$\mathcal{C}$ performance. We freeze the Encoder-Classifier (E-CL) and PE parameters optimized for joint compression and classification as described in Eq. (5.16) for AE-V_3 model, and re-trained only the decoder to minimize the MSE between $\mathbf{x}$ and $\hat{\mathbf{x}}$. A frozen E-CL and PE guarantees the same $\mathcal{R}$-$\mathcal{C}$ performance as in Fig. 5.6 with reconstruct-ability. Table 5.6 lists reconstruction PSNR and MS-SSIM values for the two test data sets at three bpp settings. The observed lower image quality, observed for the images reconstructed from the rate-accuracy optimized latents, is consistent with the reconstruction quality recorded in [31] for COCO-2017 [116] data set when the codec is optimized for joint compression and object detection.

Table 5.6: Reconstruction quality for CIFAR-10 and ImageNet-1k. L is the number of quantization centers.

Data set	L=2	L=3	L=6
		PSNR (dB)	
CIFAR-10	19.24	21.42	22.84
ImageNet-1k	20.18	21.68	22.22
		MS-SSIM	
ImageNet-1k	81.04	86.73	88.35

5.6.1 *Visualization of reconstructed images*

In our experiment, we applied a deep learning decoders similar to the architecture of [34] with modified numbers of convolution kernels to reduce the number of parameters. For CIFAR-10 reconstruction, the decoder has 0.308 M parameters which is only 44% of the ResNet-18 baseline model.



(a) CIFAR-10 (b) ImageNet-1k

Figure 5.9: Examples of (a) CIFAR-10 and (b) ImageNet-1k reconstructed images from the latent space at L =6, 3 and 2. In (b), Q=25 and Q=10 show JPEG compressed images. We observe significantly higher distortion when only 2 quantization levels are used during compression.

In Fig. 5.8(b)-(c), we provide the decoder architectures we used for CIFAR-10 and ImageNet-1k data sets. 'up' indicated the up-sampling with transposed convolution. Fig. 5.9(a) and Fig. 5.9(b) show some of the reconstructed images from 'airplane' class of CIFAR-10 [111]

and ImageNet-1k [77]. The reconstructed results look blurry compared to the original images since the MSE loss is minimized by the average of the image. We observe significantly higher distortion when only 2 quantization levels are used during compression even though the classification accuracy is less compromised (in Fig. 5.8(a)) as required.

5.7 CONCLUSIONS

In this chapter, we propose a VAE-based end-to-end framework for joint compression and classification that autonomously learns on the latent features to efficiently compress and classify images in a networked AI edge/cloud environment. Starting from the classification cross entropy loss, we present the theoretical foundation of the solution from information theory perspective to define a rate-accuracy loss similar to rate-distortion in image reconstruction. Test results of our VAE learning networks on CIFAR-10, CIFAR-100, and ImageNet-1k data sets demonstrate significant improvement of image classification accuracy at the same bit rate. We observed that a simple decoder can be trained on rate-accuracy optimized latents to synthesize visual reconstructions of the input images. However, reconstructed images displayed significant distortion at low data rates while preserving the classification accuracy. Next chapter focuses on designing learning-based codecs with improved rate-distortion-accuracy performance.

6

CLASSIFICATION-GUIDED ROI-BASED END-TO-END IMAGE COMPRESSION

As an extension of end-to-end optimization of learning-based codecs for joint image compression and classification discussed in chapter 5, this chapter focuses on designing learning-based codecs that achieve improved rate-distortion-classification ($\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$) performance simultaneously. For this purpose, we revisit the concept of region of interest (ROI) and its application in DL-based image compression[1].

In image compression applications without given ROI masks a priori, recent codecs based on deep learning have proposed effective step of ROI prediction, followed by subsequent ROI encoding [36–38]. For network-based learning over cloud, however, limited bandwidth between sensor nodes and processing servers motivates the optimization of not only $\mathcal{R}$-$\mathcal{D}$ involving ROI prediction and image encoding, but also target DL task such as classification. Existing such end-to-end ROI prediction and coding frameworks still demand heavy computation, power and memory specially during feature extraction for ROI prediction [36, 38, 117, 118]. This causes lower inference speeds and shorter battery life of sensor nodes. Targeting practical applications, in this chapter, we propose significantly lighter-weight feature extraction backbones and to guide the feature extraction process with classification cross entropy loss. By doing so, we synergies ROI compression and classification tasks demonstrating improved $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance. Our proposed joint image compression and classification framework achieves superior ROI rate-distortion and classification accuracy trade-off while maintaining good perceptual image quality. Further, our proposed classification-guided ROI codec demonstrates $\times$3.26 faster inference, 74% memory reduction and 41% power savings compared to baseline models.

[1] A part of the content of this chapter has been submitted to IEEE Transactions of Image Processing and is currently under review.

6.1 INTRODUCTION

In many practical image applications, a well-defined ROI is formed by pixels important to application specific objectives whereas the remaining pixels of less importance form "non region of interest" (NROI) for an image. For instance, the airplane in the center of Fig. 6.1 can be the ROI with the background as the NROI. Whenever possible, such applications may predefine an "ROI mask": a binary map indicating ROI areas. When such application specific ROI mask is known, commercial image compression codecs such as JPEG2000 allocate bits based on the given ROI mask [29] through a process known as "ROI coding". By allocating more encoding bits (i.e. higher rate) to ROI region over NROI, recovered images exhibit better perceptual quality with lower distortion for their ROI.

In practical IoT applications, since images collected by embedded devices are often analyzed by algorithms rather than human end users, ROI mask is not available at the image compression step. Therefore, before ROI coding, compression codec would predict potential candidates for ROI masks when they are not given. This process is known as "ROI prediction". For optimal $\mathcal{R}$-$\mathcal{D}$ performance in visualization applications, it is essential to optimize ROI prediction and ROI coding in unison [36, 38, 119], in contrast to conventional ROI applications where the optimization of ROI prediction is followed by that of ROI coding in a sequential manner [120, 121].

When training a DL-based image codec for visualization with the loss function to optimize $\mathcal{R}$-$\mathcal{D}$, existing methods lack the ability to control distortion level over regions of an image [33, 110]. Fig. 6.1 shows a clear difference of distorted areas between images reproduced by scale-hyperprior [33] codec trained with loss functions based on PSNR versus MS-SSIM. One intuitive way to overcome the ROI distortion problem is to enable image codec to learn regions/contents of higher importance. The works [36, 38] implemented such idea named "content-aware image compression" by training an auto-encoder-based codec with a learnable ROI. However, existing methods of ROI mask learning are still prohibitively

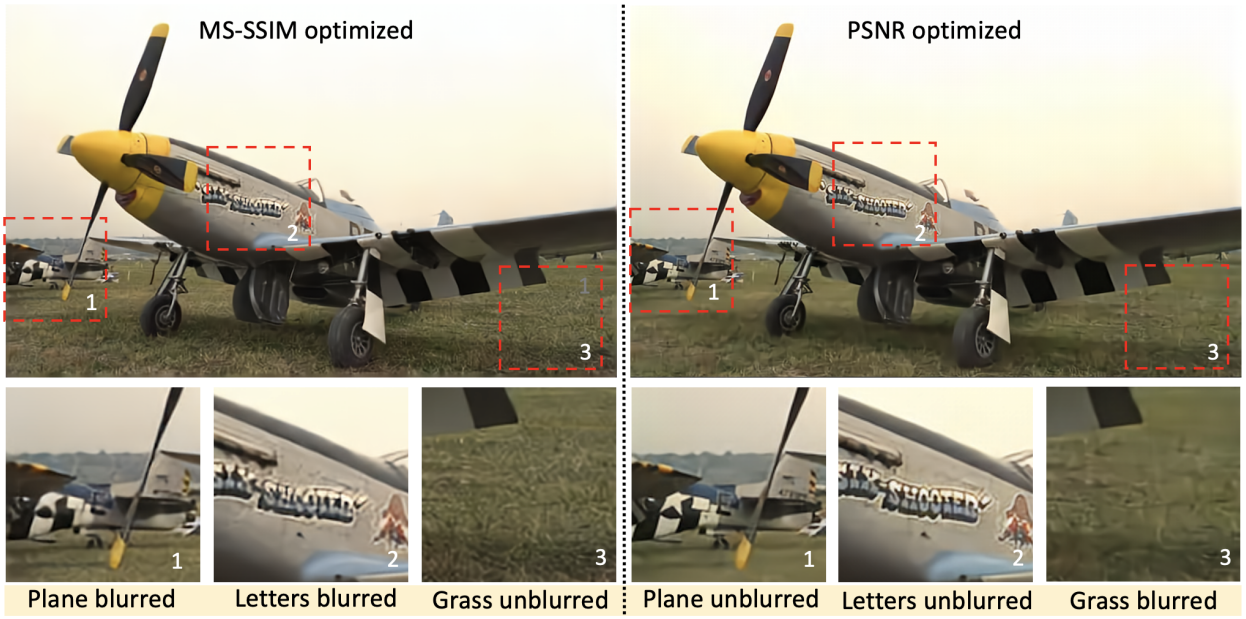


Figure 6.1: A sample of Kodak data set generated by scale-hyperprior codec optimized for MS-SSIM (left: 0.1864 bpp, PSNR=27.99, MS-SSIM=0.9803) and PSNR (right: 0.1932 bpp, PSNR=32.26, MS-SSIM=0.9713). Enlarged patches 1 and 3 indicate that MS-SSIM optimized codec adds less distortion to grass background and more distortion to the printed letters on the airplane body compared to PSNR optimized codec. The introduced distortion artifacts by the codec depend on the loss function used during training.

demanding in terms of model complexity [36, 117], memory requirement [38, 118] and power consumption [117], making them impractical to be deployed on edge devices with limited battery life.

The majority of existing ROI object detection approaches have relied on image classifier backbones for feature extraction that distinguish important and attractive objects from the image background [117, 118, 122]. In contrast to image classifier backbones, encoder backbones, while being lighter in model complexity, however weigh each pixel and their related features equally due to the nature of the distortion loss function such as PSNR or MS-SSIM. Hence, image encoder backbones in their original form, as used in [38], are less suitable for ROI prediction [31, 33].

On the other hand, the existing DL-based ROI prediction approaches are still dominated by human-vision-oriented designs [36, 38] without considering the end target applications. A vast volume of image and video content generated by the widely deployed IoT devices,

however, are commonly used for sensing and detection in data analysis by machine algorithms rather than only for human end users in visual applications. For this reason, image codec at the source device must efficiently encode features that mutually benefit data analysis and visualization applications before transmission to the resource-rich cloud for data analysis and media distribution to end users. In view of these two predominant applications, ROI mask learning should aim to improve the analytic task accuracy as well as to control rate-distortion to meet the general visualization need.

Considering image classification as the target analytical task, this paper addresses both the above issues by introducing an image-classification-guided ROI compression framework that synergies ROI compression and classification towards improved $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance. In particular, we design a faster, light-weight and power-efficient joint ROI prediction, encoding and classification framework based on recent scale-hyperprior image codec. Our contributions include 1) a novel $\mathcal{R}$-$\mathcal{D}$-optimized ROI encoding module for predefined ROI masks, 2) a light-weight saliency-map-based ROI prediction module reusing the features extracted from the shared encoder backbone, when predefined ROI masks are unavailable and, 3) an $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$-optimized joint ROI mask prediction and image encoding module that complements the specific learning task of image classification.

While achieving comparable ROI $\mathcal{R}$-$\mathcal{D}$ performance to existing DL-based ROI compression codecs [38], our proposed image codec demonstrates improved $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance compared to the state-of-the-art $\mathcal{R}$-$\mathcal{D}$ optimized codecs at significantly lower computational complexity. Further, the proposed ROI prediction module based on saliency map and feature-reuse generates fuller and more accurate ROI masks compared to ROI masks generated using deep classifier backbones [117].

We organize the rest of this chapter as follows. In Sec. 6.2, we summarize the recent approaches of ROI encoding and ROI prediction for learning-based codecs. In Sec. 6.3, we propose a novel classification-guided, end-to-end ROI prediction and coding framework, followed by an extensive experiment setup and results in Sec. 6.4. In Sec.6.5, we provide a

comparison of the model complexity, inference speed and power consumption of the proposed models. Finally, we discuss the performance benefits of the proposed feature-reusing ROI prediction module and further $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ in Sec. 6.6. See Appendix D for additional results and implementation details.

6.2 RELATED WORKS

Recent progresses in ROI-based image compression and processing have expanded from direct ROI-based coding [123, 124] followed by cascaded optimization of ROI prediction and coding [120, 121, 125] to joint optimization of ROI prediction and coding [36–38].

6.2.1 *ROI Coding*

Conventional codecs

For a known ROI mask, JPEG2000 offers ROI coding by shifting bits that encode wavelet coefficients of ROI to higher bit plane levels while downgrading encoded NROI bits to lower levels. One can prioritize ROI bits in decoding with embedded block coding through optimized truncation (EBCOT) [62]. As an alternative ROI coding method, the HEVC [96] codec changes the quantization step sizes on Discrete Cosine Transformed (DCT) coefficients of ROI. Directly utilizing the conventional JPEG2000 codec, recent works [120] and [126] proposed ROI coding under limited bandwidth for medical and communication applications. However, the non-differentiability of certain engineered blocks within the conventional codecs hinders the end-to-end adaptation of the full codec processing flow resulting sub optimal $\mathcal{R}$-$\mathcal{D}$ performance of above works [25].

Learning-based codecs

Unlike conventional codecs, most learning-based codecs are differentiable and are hence amenable to end-to-end parameter optimization based on a given loss function. Via gradient-based back-propagation of the DL network, such image codecs can optimize its encoder, decoder, and entropy coder blocks in an end-to-end manner to optimize the $\mathcal{R}$-$\mathcal{D}$ performance.

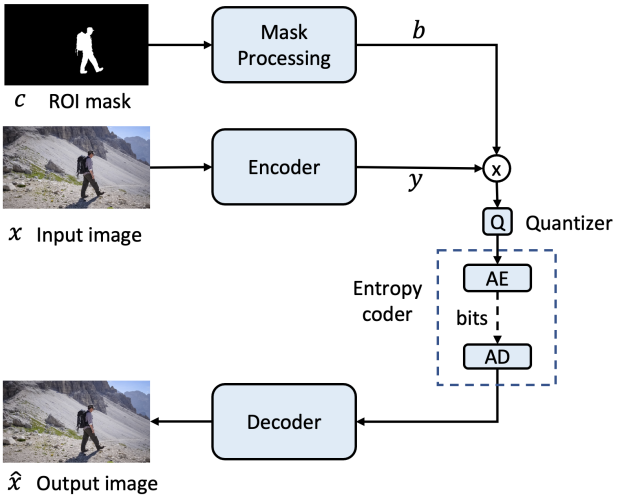


Figure 6.2: General architecture of existing ROI coding in learning based codecs. Q: Quantizer. "Mask processing" block transforms ROI mask $\boldsymbol{c}$ to $\boldsymbol{b}$ which is then element-wise multiplied by the latent representation $\boldsymbol{y}$.

Fig. 6.2 illustrates one such DL-based ROI coding approach. From the input image $\boldsymbol{x}$, important elements of latent representation $\boldsymbol{y}$ are selected according to a given ROI mask $\boldsymbol{c}$ before quantization [36, 38, 119]. This step uses element-wise product between $\boldsymbol{y}$ and the transformed ROI mask $\boldsymbol{b}$ of the same dimension as $\boldsymbol{y}$. The "mask processing" block typically consists of 2D convolution, down-sampling, and activation functions. Starting from the binary ROI mask $\boldsymbol{c}$, during the forward pass of the network, element-wise product of the transformed mask $\boldsymbol{b}$ whose elements are within [0,1], and the latent representation $\boldsymbol{y}$ allows the network to quantify the importance of elements in $\boldsymbol{y}$ by $\boldsymbol{b}$. Such element-wise product of the latent $\boldsymbol{y}$ and latent mask $\boldsymbol{b}$ weakly correlates to the element-wise product of image $\boldsymbol{x}$ and the binary mask $\boldsymbol{c}$ in spatial domain. Compared to directly stacking up the binary ROI mask $\boldsymbol{c}$ with

the input image $\boldsymbol{x}$ before the encoder [37, 125] without using explicit mask processing path, element-wise product as used in [38, 119] is more natural and interpretable. Different from recent work [119], in this chapter, we propose "multi-level mask processing": to use multiple levels of information from the mask processing path, to further quantify the importance of latent elements corresponding to ROI region.

6.2.2 *ROI Prediction*

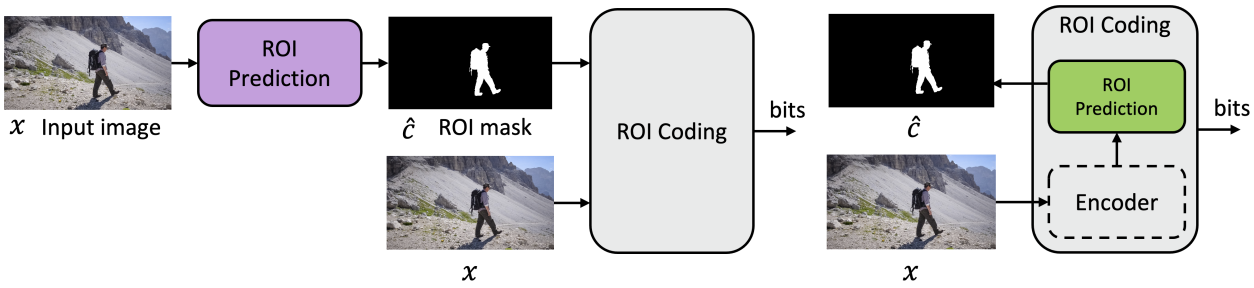
For application scenarios without known ROI masks, codec may rely on ROI prediction to generate mask candidates [36–38, 120, 121, 125]. Conventional signal processing approaches to ROI prediction include region growing [120], edge detection, and active contour segmentation [121]. These approaches are harder to integrate into an end-to-end joint optimization process and tend to yield less compelling $\mathcal{R}$-$\mathcal{D}$ performance [38].

In contrast to the such sequential ROI prediction and coding approaches, joint optimization of learning-based ROI prediction together with ROI coding can lead to more accurate ROI prediction and more efficient encoding. In this direction, the recent work of [36] has successfully utilized "saliency maps" generated from the features extracted from a VGG [13] classifier backbone for ROI prediction followed by a GAN-based image compression.

Saliency Object Detection

Saliency object detection remains a popular problem in computer vision for which recent deep learning solutions have significantly improved detection accuracy [117, 127–129]. It aims to detect a few 'important and attractive' objects on a given image [129]. For example, PoolNet [117] extracts features from different convolution layers of a deep classifier backbone before combining them to generate binary saliency maps. By aggregating lower level features from initial convolution layers for higher layer processing, PoolNet exploits multi-scale representations of objects to generate detailed masks.

For cloud-based learning applications over a network, however, it is impractical to incorporate a deep classifier backbone with large memory and computation footprint on low-cost IoT devices. Without using a dedicated classifier backbone for feature extraction during ROI prediction, the work of Cai-2019 [38] employed an implicit ROI prediction network consisting of CNN operated on multi-scale features shared with the image encoder. Such ROI prediction that shares and reuses features with the encoder is desirable to extend battery life of IoT devices. With light-weight feature sharing designs, in this chapter, we design an ROI prediction module based on saliency maps that shares the computational burden with the encoder to further simplify ROI compression. See Figure 6.3.



(a) Resource hungry ROI prediction (b) Feature-sharing ROI prediction

Figure 6.3: (a) Majority of existing ROI predictions use a deep classifier backbone for ROI prediction. (b) Proposed feature-sharing ROI prediction reuses features extracted from the encoder backbone during ROI coding.

6.2.3 *Joint image compression and classification*

Achieving $\mathcal{R}$-$\mathcal{D}$ performance in conjunction with commercial state-of-the-art codecs such as VVC [130], recent development of DL-based codes for image compression has proposed the optimization of processing flow from encoder, decoder, to entropy coding in an end-to-end manner. Furthermore, exploiting deep learning-based processing flow, recent works optimized media codec to achieve better $\mathcal{R}$-$\mathcal{C}$ performance for tasks beyond visualization by selecting appropriate optimization loss functions [25,31,87]. For instance, authors of [31] fine-tuned the

scale-hyperprior codec [33] for object detection and achieved better rate-detection accuracy by replacing image distortion loss with a detection loss function. Further, the work [87] concluded that by replacing the distortion loss by a combined distortion and classification loss function, conventional JPEG codec can be optimized for both tasks of visualization and image classification, leading to better $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance.

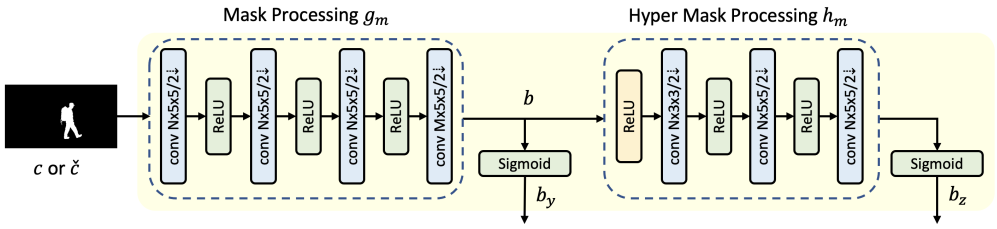
Both approaches summarized above highlight the importance of the loss function determination and the performance improvement provided by end-to-end optimization of the entire processing flow. In this chapter, we design a feature-sharing unified framework where ROI compression and image classification tasks complement each other, achieving better $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance at lower computational cost.

6.3 METHOD

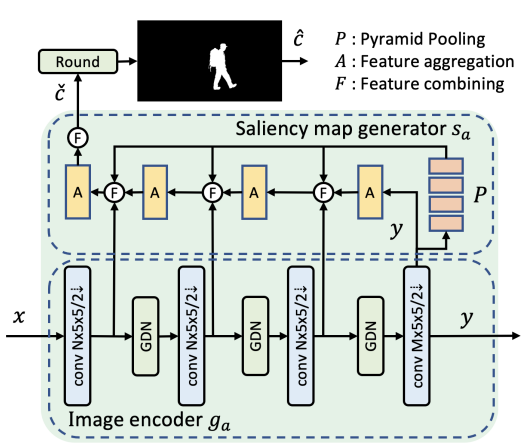
6.3.1 *ROI-based image encoding*

Consider a learning-based codec that compresses image $\boldsymbol{x} \in \mathbb{R}^D$ into a bit stream of $\mathcal{R}_y(\boldsymbol{x})$ bits. The Encoder $g_a(., \boldsymbol{\theta}_g)$ parametrized by $\boldsymbol{\theta}_g$ transforms an image $\boldsymbol{x}$ into a latent vector $\boldsymbol{y} \in \mathbb{R}^d$. The latent $\boldsymbol{y}$ is quantized into $\hat{\boldsymbol{y}}$ which is encoded by the arithmetic encoder (AE) based on entropy into a bit stream of average length $\mathcal{R}_y(\boldsymbol{x}) = \mathbb{E}_{\hat{\boldsymbol{y}}}[-\log_2 p_{\hat{\boldsymbol{y}}}(\hat{\boldsymbol{y}})]$. The decoder $g_s(., \boldsymbol{\phi}_g)$ parametrized by $\boldsymbol{\phi}_g$ reconstructs the image $\hat{\boldsymbol{x}}$ from the coded bit stream.

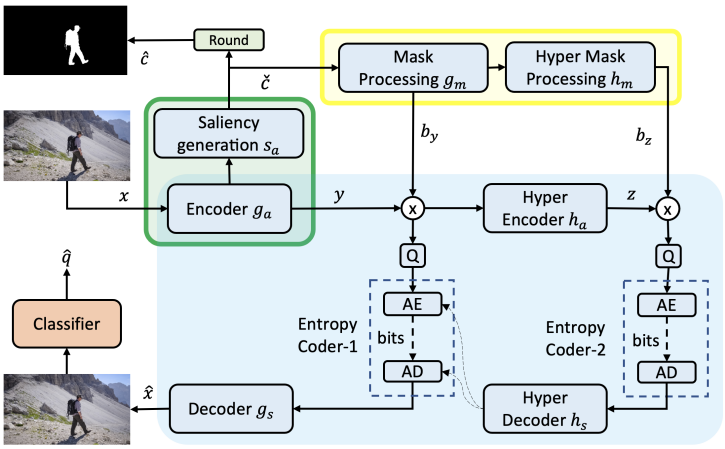
The use of scale-hyperpriors [33] further extended the above framework by encoding the variance (scale) of the latent $\boldsymbol{y}$ via another hyper encoder $h_a(., \boldsymbol{\theta}_h)$. See the blue-highlighted area in Fig. 6.4(c). A second entropy coder encodes this extra scale information (priors) $\boldsymbol{z}$ to form a separate bit stream of length $\mathcal{R}_z(\boldsymbol{x}) = \mathbb{E}_{\hat{\boldsymbol{z}}}[-\log_2 p_{\hat{\boldsymbol{z}}}(\hat{\boldsymbol{z}})]$. The total number of encoded bits equals to the sum of two bit streams: $\mathcal{R}(\boldsymbol{x}) = \mathcal{R}_y(\boldsymbol{x}) + \mathcal{R}_z(\boldsymbol{x})$. At the reconstruction stage, after decoding the bit stream of $\boldsymbol{z}$ with a hyper-decoder $h_s(., \boldsymbol{\phi}_h)$, "Entropy Coder-1" of Fig. 6.4(c) leverages the decoded priors to further decode the bit steam of $\boldsymbol{y}$.



(a) Mask processing path: we propose to use multi-level mask processing.



(b) ROI prediction reuses feature from the encoder.



(c) ROI coding and prediction for scale-hyperprior codec

Figure 6.4: (a). Mask processing path for predefined ($\boldsymbol{c}$) or estimated ($\check{\boldsymbol{c}}$) ROI predictions. (b). Saliency map generation for pre-trained scale-hyperprior encoder backbone. We use a modified PoolNet [117] architecture with pyramid pooling, feature aggregation and combining. (c). Overview of the proposed task driven approach for joint ROI image compression and classification. Saliency generation reuses the features extracted from the encoder backbone to produce the binary ROI mask. Classifier network performs classification on the reconstructed image $\hat{\boldsymbol{x}}$ to predict the class label $\hat{q}$.

For the distortion $\mathcal{D} = \mathbb{E}_{\boldsymbol{x}}[||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2]$ measured in MSE, and average rate $\mathcal{R} = \mathbb{E}_{\boldsymbol{x}}[\mathcal{R}(\boldsymbol{x})]$ in bits, the lossy image compression in scale-hyperprior codec optimizes the regularized $\mathcal{R}$-$\mathcal{D}$ objective with trade-off parameter λ for controlling the $\mathcal{R}$-$\mathcal{D}$ trade-off:

$$\boldsymbol{\theta}_g^*, \boldsymbol{\phi}_g^*, \boldsymbol{\theta}_h^*, \boldsymbol{\phi}_h^* = \arg \min_{\boldsymbol{\theta}_g, \boldsymbol{\phi}_g, \boldsymbol{\theta}_h, \boldsymbol{\phi}_h} \mathcal{R} + \lambda \mathcal{D}. \tag{6.1}$$

Training based on this objective function leads to optimized encoder parameters $(\boldsymbol{\theta}_g^*, \boldsymbol{\theta}_h^*)$ and decoder parameters $(\boldsymbol{\phi}_g^*, \boldsymbol{\phi}_h^*)$.

We can apply a mask encoder $g_m(., \boldsymbol{\psi}_g)$ to transform either the actual ROI mask $\boldsymbol{c}$ or the estimated ROI mask $\hat{\boldsymbol{c}}$ to a latent mask $\boldsymbol{b} \in \mathbb{R}^d$ to match the size of $\boldsymbol{y}$. Following the

recent works [36, 38], we element-wise multiply the latent mask $\boldsymbol{b}$ by $\boldsymbol{y}$ before quantization. Further, we apply a Sigmoid function $\tau(.)$ to limit the range of $\boldsymbol{b}_y$ to [0,1]. See Fig. 6.4(a). Hence, we write the quantized (Q) latent representation $\tilde{\boldsymbol{y}}_c$ as follows:

$$\tilde{\boldsymbol{y}}_c = Q\left(\boldsymbol{y} \cdot \tau(\boldsymbol{b})\right). \tag{6.2}$$

Utilizing multiple levels of information obtained from the scale-hyperprior codec, we generate multi-level mask processing by reusing the latent mask $\boldsymbol{b}$ to match the dimension of hyperprior $\boldsymbol{z}$. As shown in Fig, 6.4(a), hyper mask encoder $h_m(., \boldsymbol{\psi}_h)$ followed by Sigmoid function $\tau(.)$ transforms the latent mask to $\boldsymbol{b}_z$. Similar to (6.2), we generate new priors $\tilde{\boldsymbol{z}}_c$ as follows:

$$\tilde{\boldsymbol{z}}_c = Q\left(\boldsymbol{z} \cdot \tau(h_m(\boldsymbol{b}, \boldsymbol{\psi}_h))\right). \tag{6.3}$$

We denote the sum rate of new bit stream given ROI mask $\boldsymbol{c}$ as $\mathcal{R}_c(\boldsymbol{x}) = \mathcal{R}_{\boldsymbol{y}_c}(\boldsymbol{x}) + \mathcal{R}_{\boldsymbol{z}_c}(\boldsymbol{x})$. For a given image $\boldsymbol{x}$ and ROI mask $\boldsymbol{c}$, we apply an approach similar to those from [36, 38] to define the loss function of $\mathcal{R}$-$\mathcal{D}$ objective for ROI based image compression as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{c}) = \mathcal{R}_c(\boldsymbol{x}) + \lambda \mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda_{\text{roi}} \mathcal{D}_{\text{roi}}(\boldsymbol{x} \cdot \boldsymbol{c}, \hat{\boldsymbol{x}} \cdot \boldsymbol{c}) \tag{6.4}$$

$\mathcal{D}$ and $\mathcal{D}_{\text{roi}}$ measure the perceptual quality loss of the full image and the ROI region, respectively, and are weighted by regularizers λ and λ_{roi}. Similar to [36–38, 125], we let $1 - \text{MS-SSIM}$ be loss for $\mathcal{D}$, and MSE be loss for $\mathcal{D}_{\text{roi}}$. Through end-to-end optimization to minimize the cumulative loss function of (6.4), the parameters of encoder $(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h)$, decoder $(\boldsymbol{\phi}_g, \boldsymbol{\phi}_h)$, and the mask encoder $(\boldsymbol{\psi}_g, \boldsymbol{\psi}_h)$ can be jointly optimized as

$$\overbrace{\boldsymbol{\theta}_g^*, \boldsymbol{\theta}_h^*, \boldsymbol{\phi}_g^*, \boldsymbol{\phi}_h^*}^{\text{codec params.}}, \quad \overbrace{\boldsymbol{\psi}_g^*, \boldsymbol{\psi}_h^*}^{\text{mask processing params.}} = \arg \min_{\boldsymbol{\theta}_g, \boldsymbol{\phi}_g, \boldsymbol{\theta}_g, \boldsymbol{\phi}_h, \boldsymbol{\psi}_g, \boldsymbol{\psi}_h} \mathcal{R}_c + \lambda \mathcal{D} + \lambda_{\text{roi}} \mathcal{D}_{\text{roi}}. \tag{6.5}$$

Here, $\mathcal{R}_c$, $\mathcal{D}$ and $\mathcal{D}_{\text{roi}}$ denote the data rate, overall image distortion, and ROI distortion averaged over the samples $\boldsymbol{x}$, respectively, for an ROI mask $\boldsymbol{c}$.

6.3.2 *ROI prediction with Saliency maps*

In most cloud-based image applications, ROI masks are not available to ROI coding [36, 38]. Inspired by the success reported in [36], we apply CNN-based saliency maps to derive ROI mask candidates directly from input images. The distinct feature of our design lies in the removal of a separate feature extraction network to generate saliency map as in [36]. Instead, for efficiency and complexity reduction, we propose to reuse the multi-scale features already acquired by the encoder backbone, as conceptually shown in Fig. 6.4(b).

When designing a novel saliency map generation module on top of the scale-hyperprior encoder backbone, we incorporate a PoolNet [117] which was originally designed for classification backbones such as ResNet [15]. For a given RGB image $\boldsymbol{x} \in \mathbb{R}^{w \times h \times 3}$ of width w and height h, our proposed saliency map generator uses 4 levels of features **extracted** by the encoder backbone as inputs, as seen from Fig. 6.4(b). Starting from latent $\boldsymbol{y}$, the "pyramid pooling module" (P) **interpolates** $\boldsymbol{y}$ to match the spatial dimension of the remaining 3 feature levels. Before combining, "feature aggregation module" (A) uses 2D convolution to further process and to **aggregate** features from the previous level. We then apply a "feature combining module" (F) to combine features after extraction, interpolation, and aggregation at each level. Finally, saliency generator $s_a(., \boldsymbol{\eta}_s)$ produces the score map $\check{\boldsymbol{c}} \in \mathbb{R}^{w \times h}$ as output vector whose elements are in [0,1]. Refer [129] for the detailed implementations of P, A and F. As shown in Fig. 6.4(a), we send the generated saliency map without rounding ($\check{\boldsymbol{c}}$) to the mask generator to facilitate gradient back propagation. We only apply rounding to generate the binary ROI map $\hat{\boldsymbol{c}}$ from $\check{\boldsymbol{c}}$ when evaluating the distortion of the ROI region $\mathcal{D}_{\text{roi}}$.

6.3.3 *Classification-guided ROI compression*

In end-to-end optimization of any learning-based work flow, loss function plays a vital role. For joint ROI prediction and ROI coding frameworks without a given ROI mask $\boldsymbol{c}$, ROI prediction network can generate $\hat{\boldsymbol{c}}$ through an end-to-end training and optimization [38].

In contrast to image classifier backbones, encoder backbones weigh each pixel and their related features equally due to the nature of the distortion loss function such as PSNR or $1 - \text{MS-SSIM}$. Hence, image encoder backbones in their original form are less suitable for saliency object detection [31, 33]. We proposes to replace the computationally-heavy classifier backbone with a light-weight encoder backbone during ROI, and make the encoder learn the features that aid in saliency object detection. Our idea is to guide ROI prediction based on image classification outcomes, while optimizing the entire ROI prediction and coding network for $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance. To this end, we adopt a pre-trained, off-the-shelf classifier $r(., \boldsymbol{\kappa}_r)$ on reconstructed images $\hat{\boldsymbol{x}}$ to generate image label $\hat{q}$ as shown in Fig. 6.4(c). Not relying on a customized classifier enables easy and practical use of our proposed codec with existing applications without having to redesign/retrain classifier models at the cloud-end [30].

We formulate the loss function of joint ROI compression and classification task for training on each pair of data {input image $\boldsymbol{x}$, ground truth class label q_{gt}} as follows. Similar to Eq. (6.4), $\mathcal{R}$-$\mathcal{D}$ loss of ROI compression for the predicted mask $\hat{\boldsymbol{c}}$ is:

$$\mathcal{L}_{\text{RD}}(\boldsymbol{x}) = \mathcal{R}_{\hat{c}}(\boldsymbol{x}) + \lambda_{\text{ssim}} \cdot \mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda_{\text{roi}} \cdot \mathcal{D}_{\text{roi}}(\boldsymbol{x} \cdot \hat{\boldsymbol{c}}, \hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{c}}). \tag{6.6}$$

Here, the first element $\mathcal{R}_{\hat{c}}(\boldsymbol{x})$ is the sum bit rate that includes the quantized $\tilde{\boldsymbol{y}}_{\hat{c}}$ and $\tilde{\boldsymbol{z}}_{\hat{c}}$ as discussed with respect to Eqs. (6.2) and (6.3). We also define image classification loss

according to the cross entropy between the estimated class label $\hat{q}$ and ground truth class label q_{gt}.

$$\mathcal{L}_{\text{CL}}(\boldsymbol{x}) = \text{CE}(q_{\text{gt}}, \hat{q}) \tag{6.7}$$

As proposed in [129], we further include a cross entropy loss to measure the accuracy of the predicted saliency mask $\hat{c}$ in comparison with the true mask c.

$$\mathcal{L}_{\text{S}}(\boldsymbol{x}) = \text{CE}(\boldsymbol{c}, \hat{\boldsymbol{c}}) \tag{6.8}$$

Incorporating the above task losses, we directly formulate a joint end-to-end loss $\mathcal{L}_J$ through regularizing control parameters α and β:

$$\mathcal{L}_{\text{J}}(\boldsymbol{x}) = \mathcal{L}_{\text{RD}}(\boldsymbol{x}) + \alpha \cdot \mathcal{L}_{\text{CL}}(\boldsymbol{x}) + \beta \cdot \mathcal{L}_{\text{S}}(\boldsymbol{x}) \tag{6.9}$$

to optimize the proposed joint image compression and classification framework by minimizing the mean joint loss via

$$\left\{ \overbrace{\boldsymbol{\theta}_g^*, \boldsymbol{\theta}_h^*, \boldsymbol{\phi}_g^*, \boldsymbol{\phi}_h^*}^{\text{codec}}, \overbrace{\boldsymbol{\psi}_g^*, \boldsymbol{\psi}_h^*}^{\text{mask processing}} , \overbrace{\boldsymbol{\eta}_s^*}^{\text{saliency}} \right\} = \min_{\boldsymbol{\theta}_g, \boldsymbol{\phi}_g, \boldsymbol{\theta}_g, \boldsymbol{\phi}_h, \boldsymbol{\psi}_g, \boldsymbol{\psi}_h, \boldsymbol{\eta}_s} \mathbb{E}_{\boldsymbol{x}}[\mathcal{L}_{\text{J}}(\boldsymbol{x})]. \tag{6.10}$$

6.4 EXPERIMENTS AND RESULTS

We first introduce the training setup of our experiments before presenting the results.

6.4.1 *Details on Training*

ROI coding

We train the proposed ROI coding module on HKU-IS data set based on known ROI mask. In this setting, we train to optimize parameters of the "Mask Processing" path of Fig. 6.4(a) and the scale-hyperprior codec shown as blue region of Fig. 6.4(c) according to Eq. (6.5). We initialize scale-hyperprior codec parameters $(\boldsymbol{\theta}_g, \boldsymbol{\phi}_g, \boldsymbol{\theta}_h, \boldsymbol{\phi}_h)$ with off-the-shelf, pre-trained model available at [131]. Starting from the learning rates of 5e-5 for the codec and 1e-3 for the mask processing path, we train the entire network with an "Adam" optimizer for 20 epochs while reducing learning rates to 1/10 at epochs 10 and 15.

In order to obtain different rate-distortion points, we start with an expected rate r_{T} in bits per pixel (bpp). We then constrain the rate $\mathcal{R}$ by applying a ReLU() function in

$$\min \mathrm{ReLU}(\mathcal{R} - r_{\mathrm{T}}) + \lambda \cdot \mathcal{D} + \lambda_{\mathrm{roi}} \cdot \mathcal{D}_{\mathrm{roi}} \tag{6.11}$$

and vary the trade-off parameters λ and λ_{roi} accordingly.

Classification-guided ROI compression

In this framework, we optimize the full processing flow shown in Fig. 6.4(c), according to Eq. (6.10) in an end-to-end manner. For the classifier, we adopted an available ResNet-18 model which has been pre-trained for ImageNet-1k. We freeze the classifier at all time and and use it to compute the classification cross entropy loss $\mathcal{L}_{\mathrm{CL}}$ in Eq. (6.7).

Once the ROI coding module was trained for given ROI masks as explained in Sec. 6.4.1, we initialize the scale-hyperprior codec $(\boldsymbol{\theta}_g, \boldsymbol{\phi}_g, \boldsymbol{\theta}_h, \boldsymbol{\phi}_h)$ and the mask processing path $(\boldsymbol{\psi}_g, \boldsymbol{\psi}_h)$ using the results learned in Sec. 6.4.1. Starting from initial learning rates of 5e-5 for the scale-hyperprior codec, 1e-3 for the mask processing path, and 5e-5 for the saliency map generation unit, we apply "Adam" optimizer for 20 epochs by reducing the learning rate to

1/10 of the original at the epochs 10 and 15. Further, we controlled the data rate using the same strategy in Sec. 6.4.1 and changed the trade-off parameters λ_{ssim}, λ_{roi}, α and β accordingly.

Since the ground truth (GT) ROI masks c in Eq. (6.8) are not available for classification data sets in general, we adopt a PoolNet [117] with ResNet-50 backbone trained on HKU-IS data set to generate binary masks. We found the PoolNet model to generate accurate ROI masks very close to ground truth with a mean absolute error (MAE) of only 0.0362, which is comparable to the test accuracy reported in [129]. Please refer to Appendix D for a visual comparison of PoolNet-generated masks for HKU-IS test set. Recall that the objective of Eq. (6.8) is to guide ROI mask generation and not to strictly learn to generate exact ROI masks as ground truths. Hence, we apply a smaller value β for saliency loss during training. This light control of mask learning led to more complete and accurate ROI masks, as will be shown in Sec. 6.4.2.

Performance evaluation

We evaluated performance of the proposed ROI coding module in terms of PSNR (dB) of the ROI region and the overall image perceptual quality measured in MS-SSIM. Similar to [38], we convert MS-SSIM into dB via $10\log_{10}(1 - \text{MS-SSIM})$. When calculating the ROI PSNR for HKU-IS and DAVIS-2016 data sets, we used available ground truth masks and compared the performance against two benchmarks: the end-to-end ROI compression framework (Cai-2019) [38] and standard JPEG2000 ROI compression.

In order to further establish the robust performance of the proposed ROI compression over different mask sizes, we tested the proposed ROI coding module trained on the HKU-IS data set in Sec. 6.4.1, on the DAVIS-16 data set.

To evaluate the proposed framework of joint ROI compression and classification, we further record image classification accuracy in-addition to the rate-distortion ($\mathcal{R}$-$\mathcal{D}$) metrics. Since the learned ROI predictions within our proposed method are better than PoolNet-generated

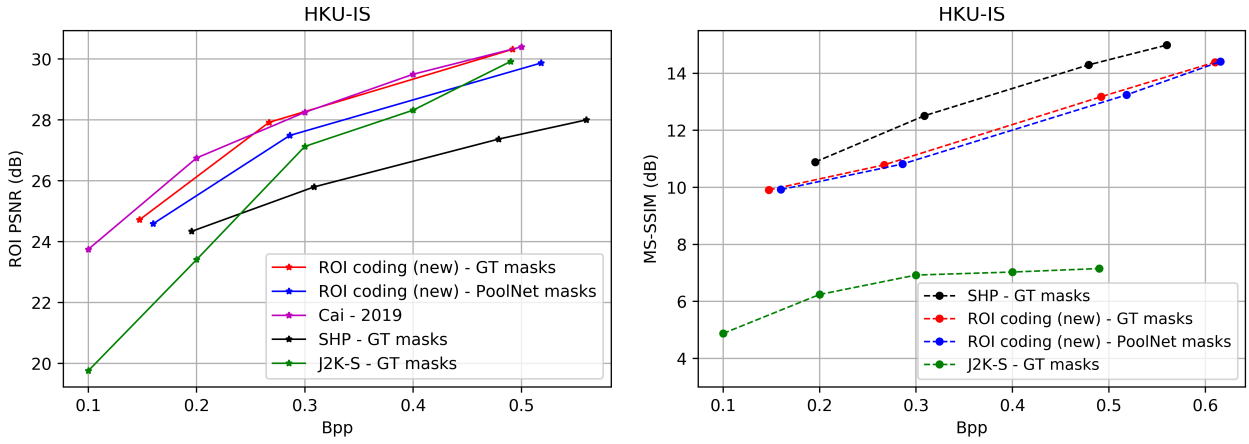
saliency masks, we provide the ROI PSNR for the proposed methods based on the learned predictions.

6.4.2 *Results and Comparisons*

ROI coding with predefined ROI regions

Fig. 6.5 provides the $\mathcal{R}$-$\mathcal{D}$ performance of the proposed ROI coding module over HKU-IS test data. Fig. 6.5(a) presents the resulting PSNR for ground truth (GT) ROI by comparing scale-hyperprior (SHP) codec without the ROI feature, JPEG2000, Cai-2019 [38], and our proposed method. When evaluating the SHP codec, we used the off-the-shelf MSE optimized models available at [131]. For JPEG2000 evaluation, we used "scaling"-based ROI compression [29] available in Kakadu[2] implementation of JPEG2000. In addition, we included the ROI PSNR performance of the proposed codec when evaluating under PoolNet-generated saliency masks (See Fig. D.1). Note that the proposed method matches the performance of Cai-2019, while our proposed classification-guided ROI codec showcases a 73% lighter ROI prediction

[2] https://kakadusoftware.com



(a) ROI PSNR (b) Overall MS-SSIM

Figure 6.5: Rate-distortion results for HKU-IS data set. (a) PSRN comparisons of the ground truth (GT) ROI region for: scale-hyperprior codec (SHP) without the ROI feature, JPEG2000, Cai-2019 [38] and our proposed method: ROI coding (new). (b) comparison of full image quality measured in MS-SSIM.

backbone resulting an overall 18% lighter model in terms of number of parameters compared to Cai-2019 as shown in Sec. 6.5. Further, the proposed codec outperforms both JPEG2000 and SHP codec.

In order to show the effect on overall image quality while achieving better ROI performance, we provide MS-SSIM for the HKU-IS test set in Fig. 6.5(b). We see that the proposed method outperforms JPEG2000 codec by a significant margin.

Fig. 6.6 illustrates the performance of the proposed codec on DAVIS-2016 data set for various mask sizes. Our proposed solution achieves better $\mathcal{R}$-$\mathcal{D}$ performance than SHP in terms of ROI PSNR, as Fig. 6.6(a) shows. We also note larger gains for smaller ROI mask sizes because the codec allocates fewer additional bits to encode features of ROI regions in such cases. Fig. 6.6(b) compares MS-SSIM of the full image. The results show that, the improved ROI PSNR for various test mask sizes is accompanied by a small marginal loss of MS-SSIM for the full image in comparison with SHP codec.

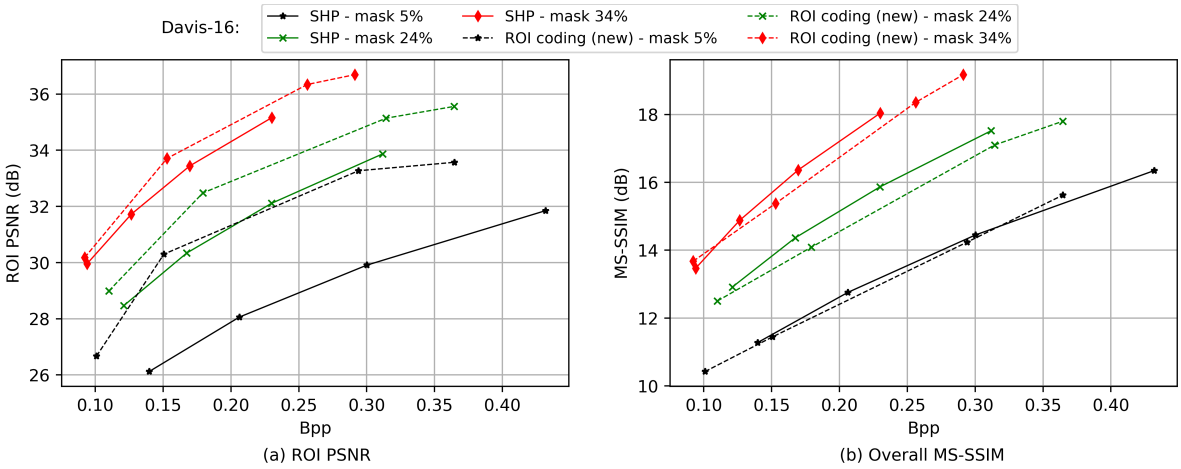


Figure 6.6: $\mathcal{R}$-$\mathcal{D}$ results for different mask sizes of DAVIS-2016 data set. (a) compares ROI PSNR values and (b) compares the overall image quality measured in MS-SSIM with scale-hyperprior (SHP) off-the-shelf models. We observe better ROI $\mathcal{R}$-$\mathcal{D}$ performance in terms of PSNR and larger gains for smaller ROI mask compared to bigger mask sizes while maintaining comparable overall image quality.

Joint ROI compression and classification

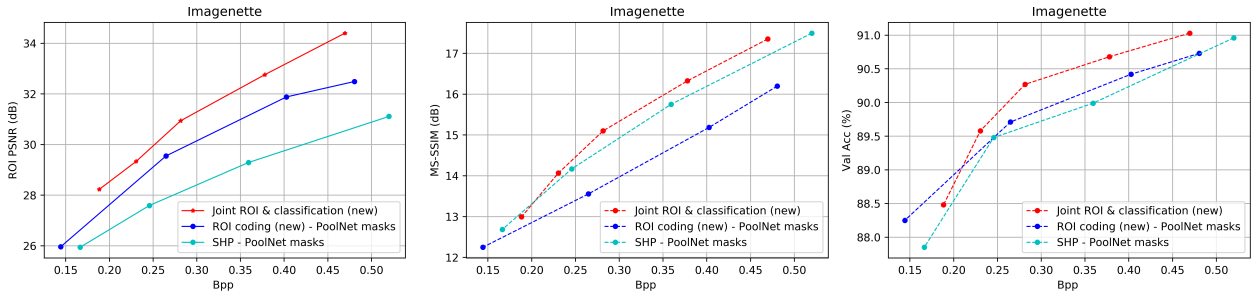
We evaluated the performance of proposed joint ROI compression and classification framework of Fig. 6.4 that utilizes pretrained off-the-shelf image classifiers on four data sets: Imagenette, CADDY, ASL and ImageNet-1k.

Fig. 6.7 shows $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance for Imagenette. To obtain an off-the-shelf classier for Imagenette with 10 classes, we started with a ResNet-18 classifier pre-trained for ImageNet-1k but altered the fully connected layer to classify 10 classes. We then fine-tuned the modified ResNet-18 classifier on Imagenette training set. In Fig. 6.7(a), we compared the ROI PSNR performance of our proposed framework against the SHP codec as baseline. To ensure fairness, we also fine-tuned the SHP codec to optimize $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ on Imagenette data set by applying the optimization step similar to the proposal of [31]:

$$\min_{\boldsymbol{\theta}_g,\boldsymbol{\phi}_g,\boldsymbol{\theta}_h,\boldsymbol{\phi}_h} \mathcal{R} + \lambda \cdot \mathcal{D} + \alpha \cdot \mathcal{L}_{\text{CL}} \tag{6.12}$$

By minimizing the above loss function for the training set, we fine-tune the SHP codec to reproduce images with higher visual quality and better classification accuracy.

In addition to the proposed joint compression and classification framework with learned masks, we further trained the proposed ROI coding module given the PoolNet-generated masks as ground truth (marked blue in Fig. 6.7(a)). Similarly for fairness, we also fine-tuned



(a) ROI PSNR (b) Overall MS-SSIM (c) Classification accuracy

Figure 6.7: $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ results for Imagenette data set. The proposed joint ROI prediction, coding and classification framework (red) achieves better ROI PSNR, overall image quality and classification accuracy.

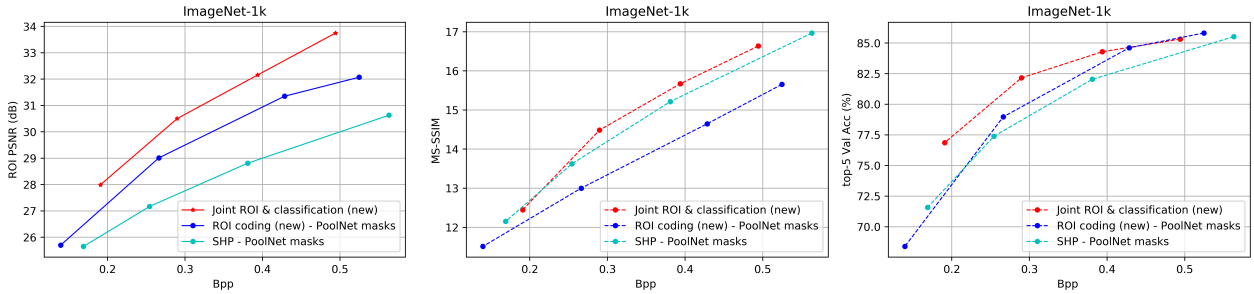
the ROI coding module by excluding the saliency loss term in Eq. (6.9), since ROI masks learning is not required.

$$\mathcal{L}_{\mathrm{J}}(\boldsymbol{x}) = \mathcal{L}_{\mathrm{RD}}(\boldsymbol{x}) + \alpha \cdot \mathcal{L}_{\mathrm{CL}}(\boldsymbol{x}) \tag{6.13}$$

When calculating the ROI PSNR for the baseline SHP codec and the ROI coding module, we applied PoolNet saliency masks. We observe that our proposed classification-guided mask-learning delivers significant gains in ROI PSNR in these two cases. Equally noteworthy, the proposed method achieves higher overall image quality and classification accuracy. These results establish that classification-guided ROI mask learning, and end-to-end optimization of ROI compression are both valuable design principles for performance enhancement.

Evaluation on ImageNet-1k data set

We further evaluate the performance of the proposed codec trained on Imagenette data set described in Sec. 6.4.2 on the much larger classification data set of ImageNet-1k. To do so, we simply replace the classifier used for Imagenette with the original off-the-shelf ResNet-18 classifier for 1000 classes. We directly test this model on ImageNet-1k without any fine-tuning. Similarly, we also obtained the performance of the baseline SHP codec and the ROI compression module by directly using the models trained for Imagenette data in



(a) ROI PSNR (b) Overall MS-SSIM (c) Classification accuracy

Figure 6.8: $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ results for ImageNet-1k data set. We pre-trained the codecs on Imagenette data set and evaluated on ImageNet-1k test set. An off-the-shelf ResNet-18 classifier was used to evaluate the classification accuracy.

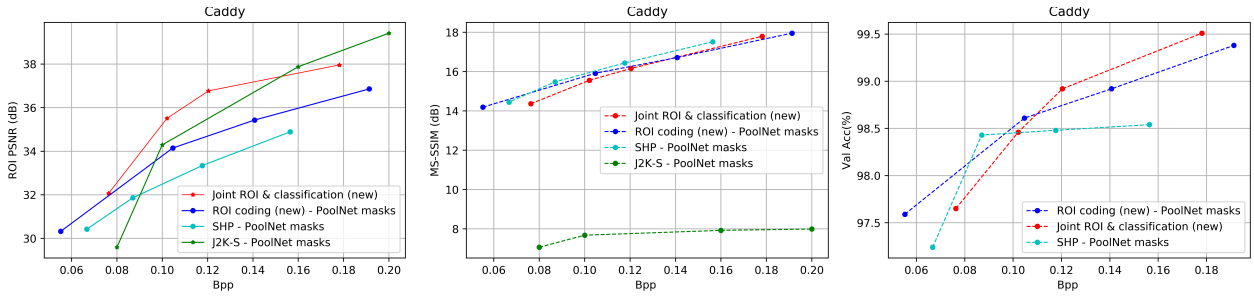
Sec. 6.4.2 on ImageNet-1k. We recorded top-5 accuracy as the classification accuracy. Fig. 6.8 presents the results.

Despite the mismatch of training data versus testing data, the proposed codec outperforms the baselines in terms of ROI PSNR, overall MS-SSIM, and classification accuracy. This experiment setup further suggests that once the codec has been optimized for classification on one data set, similar performance gains can be achieved on similar data sets. Such benefit is derived from the use of cross entropy loss and off-the-shelf classifiers to guide ROI prediction and ROI coding modules without retraining the classifier. As such, the proposed codec learns to extract necessary features for ROI prediction, coding, and classification without being overly reliant on the classification data set.

Underwater gesture recognition

Using the CADDY data set, we test the proposed codec by evaluating on a less well known but practical application of underwater gesture recognition. The task is to classify underwater gesture signals for communication into one of 16 classes: "end comm, up, mosaic, five, here, backwards, two, boat, four, start comm, down, photo, one, num delimiter, carry, three". Such images mostly consist of a diver gesturing a signal with the ROI region being the diver.

Similar to our Imagenette and ImageNet-1k experiments, we use PoolNet-generated masks as ground truth. To train and evaluate the proposed method over the CADDY data, we



(a) ROI PSNR (b) Overall MS-SSIM (c) Classification accuracy

Figure 6.9: $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance of the proposed joint image compression and classification framework for CADDY data set. The proposed joint ROI prediction, coding and classification framework (red) achieves better ROI PSNR, classification accuracy and sufficient overall image quality.

Figure 6.10: Generated masks (columns: 3-6) at different data rates for CADDY data set where ground truth masks are incomplete and less accurate. Note that we demonstrate the samples with worst ground truth masks generated by PoolNet [117]. Reconstructed images $\hat{\boldsymbol{x}}$ are shown in the last column.

followed the same training and fine-tuning steps performed for Imagenette classification of Sec. 6.4.2. Fig 6.9 provides the $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance. The proposed end-to-end ROI prediction and coding method delivers superior ROI PSNR and full image quality over a range of viable data rate. It outperforms the baseline SHP codec and ROI compression at modestly higher data rate (bpp). In addition, we further tested ROI $\mathcal{R}$-$\mathcal{D}$ performance for JPEG2000 codec for comparison. When generating this JPEG2000 benchmark, we followed the same steps described in Sec. 6.4.2 for HKU-IS data set.

We show the learned ROI masks generated by the proposed framework at different bit rates in Fig. 6.10. Compared to the ground truth masks generated by PoolNet (pre-trained on HKU-IS data set), our leaned masks are more accurate. Note that for this illustration, we compare the worst ROI masks generated by PoolNet. Further, in column 7 of Fig. 6.10, we show the reconstructed images based on the learned masks at an average Bpp of 0.1781. Note

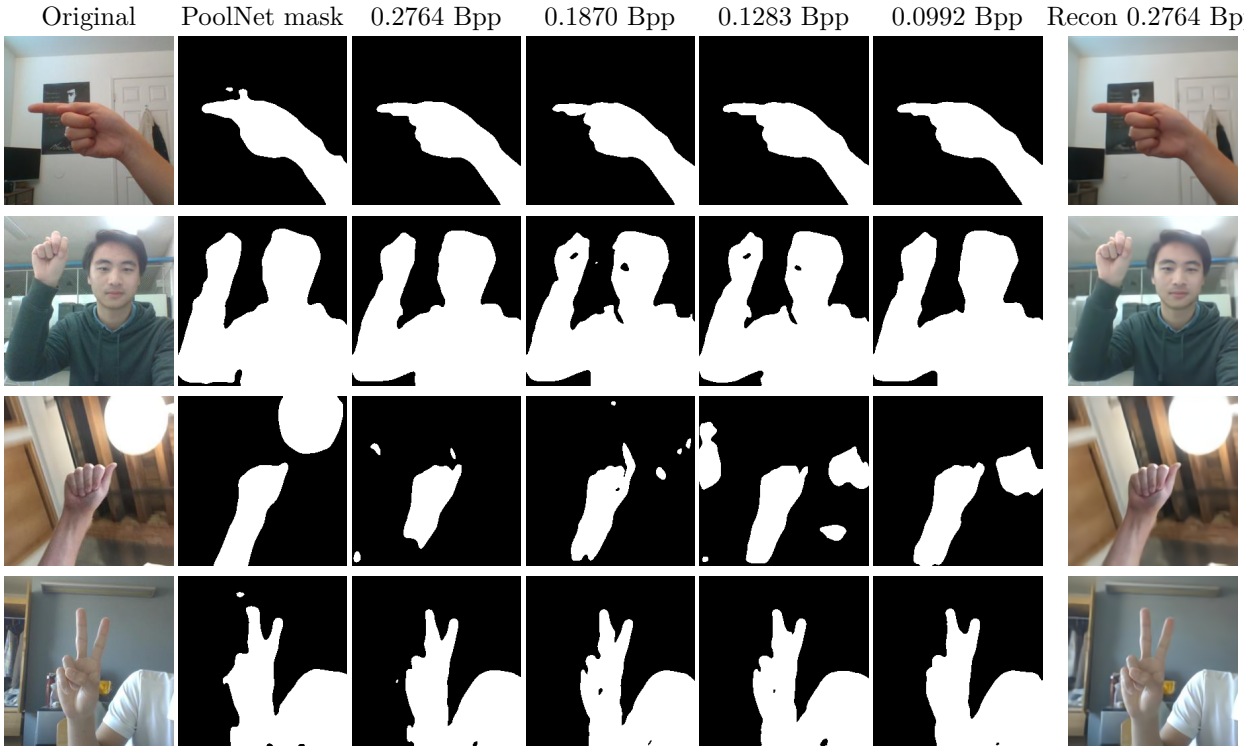


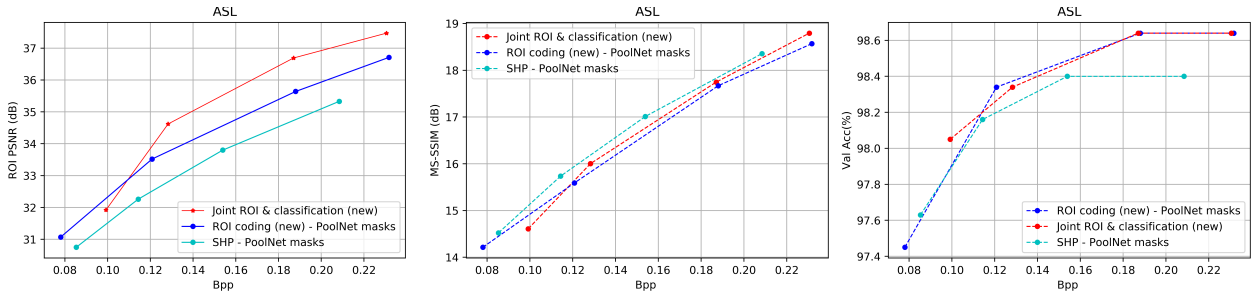
Figure 6.11: Learned masks for ASL data set. Note that the learned masks are more detailed and accurate compared to the ground truth masks generated by PoolNet [117]. Reconstructed images $\hat{\boldsymbol{x}}$ are shown in the last column.

that the proposed method tends to segment the masks following color gradient differences specially at higher bit rates as opposed to lower Bpps.

American Sign Language character recognition

We further tested the proposed joint ROI compression codec with a more common and practical classification application. A sample image of the data set depicts a person gesturing a hand signal in a moderately clear background. Hence, the ROI region is the person gesturing the sign. The task is to classify each hand sign to one of 24 English characters.

During training, we followed the exact same steps implemented for CADDY underwater gesture recognition task and similarly, Fig. 6.12 shows $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance for the test set. In consistence with Imagenette, ImageNet-1k and CADDY experiments, the proposed classification guided ROI compression codec achieves higher ROI PSNR and classification



(a) ROI PSNR (b) Overall MS-SSIM (c) Classification accuracy

Figure 6.12: $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance of the proposed joint image compression and classification framework for ASL data set. The proposed joint ROI compression and classification framework (red) achieves better ROI PSNR, classification accuracy and sufficient overall image quality.

accuracy while maintaining sufficient overall image quality. Further, in Fig. 6.11, we compare the learned ROI predictions at different bit rates. Note that the learned masks via the proposed method in column 3 are more detailed and accurate compared to PoolNet generated ground truths (column 2). This further solidifies the observations we noted for CADDY data set.

6.5 COMPLEXITY COMPARISON

In this section, we compare the model complexity, inference speed and power consumption of the proposed classification-guided ROI codec against recent learning-based ROI codecs.

6.5.1 *Model complexity*

Recall that we proposed to learn ROI masks by using the features extracted from the shared encoder backbone on resource-limited embedded devices, when predefined ROI predictions are not available for networked learning applications. This solution overcomes the need for more memory to store a deep classifier backbone (e.g. ResNet-50) and computation power

required to estimate features by the conventional saliency based ROI prediction approach consisting of a separate path for mask estimation [36, 119].

Table 6.1: Complexity comparison: No. of parameters are given in millions (M). Our proposed classification-guided approach shows 74% (3 times) of parameter reduction compared to the baseline. Parameters are categorized in to feature extraction backbone (B.bone) and saliency mask generation (Mask) under ROI prediction, and main codec backbone (Codec) and mask processing (Mask) under ROI coding.

	ROI prediction		ROI coding		
Method	B.bone	Mask	Codec	Mask	Total
PoolNet + SHP (a)	25.51	42.75	5.08	-	73.34
Cai-2019[38] (estimated)	5.45	12.30	5.45	-	23.21
PoolNet + ROI coding (b)	25.51	42.75	5.08	2.48	75.82
Classification-guided ROI	**1.49**	**11.41**	**3.59**	2.48	**18.97**
Complexity reduction					
– compared to (a)	94%	73%	29%	-	74%
– compared to (b)	94%	73%	29%	0%	75%
– compared to Cai-2019	73%	7%	34%	-	18%

In Table. 6.1, we calculate the network complexity in terms of number of parameters. Note that the proposed feature-sharing classification-guided ROI codec uses a significantly simple ROI prediction (feature extraction) backbone with 94% of reduced complexity compared to PoolNet while achiving superior ROI compression performance shown in Figures 6.7, 6.8, 6.9 and 6.12. Overall, with 73% reduction in saliency mask generation, 29% in ROI coding, classification-guided ROI codec records 74% total complexity reduction compared to PoolNet-based ROI codecs in terms of no. of parameters. Detailed complexity comparisons can be found in the supplement material.

Compared to Cai-2019, our proposed ROI prediction backbone is 73% lower in number of parameters resulting an overal complexity reduction of 18%. When calculating number of parameters for Cai-2019 [38], we relied on the implementation details provided in their paper since implementation code is unavailable. See Appendix D for the details of our parameter estimation of Cai-2019.

6.5.2 *Inference speed*

We tested the inference speed of the proposed and the baseline codecs in terms of images per second (ips) on an Nvidia Titan V GPU based on CADDY test set. See Table. 6.2 for results. Speed gains were calculated compared to the baseline (PoolNet + SHP) which uses PoolNet for ROI prediction and SHP codec for image coding. For fairness, images were resized to 256×256 before inference.

Combining our ROI coding module with SHP codec (PoolNet + ROI coding (b) in Table. 6.2) introduces only 23% reduction of inference speed from SHP codec. On the other hand, our classification-guided ROI codec shows 3.36 times inference speed compared to the baseline PoolNet + SHP, benefiting from the reduced complexity as shown in Sec. 6.5.1.

Table 6.2: Inference speed comparison: Speed is given in number of images per second (ips). Our proposed feature-sharing, classification-guided ROI prediction shows 5.38 times faster inference compared to PoolNet.

Method	ROI prediction	ROI coding	Total
PoolNet + SHP (a)	32.73	**264.00**	29.10
PoolNet + ROI coding (b)	32.73	205.33	28.21
Classification-guided ROI	**176.00**	205.33	**94.76**
Speed gain			
– compared to (a)	×5.38	×0.77	×3.26
– compared to (b)	×5.38	×1	×3.36

6.5.3 *FLOPS calculation*

When measured on the same device (GPU) the number of floating point operations per second (FLOPS) is directly proportional to the power consumption. In Table. 6.3, we compare the number of FLOPS required for the baselines and the proposed ROI codec on an Nvidia Titan V GPU. Demonstrating similar trends to complexity savings, the proposed classification-guided ROI codec records 47% power savings during ROI prediction and 41% overall compared to PoolNet + SHP baseline.

Table 6.3: Computational power demand in terms of Flops in Billions (G).

	ROI prediction		ROI coding		
Method	B.bone	Mask	Codec	Mask	Total
PoolNet + SHP (a)	8.77	41.90	5.71	-	55.38
PoolNet + ROI coding (b)	8.77	41.90	5.71	2.50	57.88
Classification-guided ROI	**2.76**	**24.32**	**2.94**	2.50	**32.52**
Power savings					
– compared to (a)	69%	42%	49%	-	41%
– compared to (b)	69%	42%	49%	0%	44%

6.6 DISCUSSION: LEARNED MASKS FROM SHARED ENCODER BACKBONE

In order to visualize the benefits of classification-guided saliency predictions, we perform the following ablation test. We train the proposed saliency generation module $s_a(., \boldsymbol{\eta}_s)$ on the SHP encoder which has been pre-trained with (SHP-Q8 optimized) and without (SHP-Q8) a classification loss function. For both settings, we used SHP codec with the saliency generation path as given in Fig 6.4(c) without the mask processing path and the classifier. Our goal is to assess the performance of the proposed saliency generator.

SHP-Q8

For this setting, we initialize SHP codec with highest available quality (Q=8) mode and froze the codec during training. We optimize parameters $\boldsymbol{\eta}_s$ of s_a through training on HKU-IS to minimize the saliency loss of Eq (6.8). During training, we applied Adam optimizer for 40 epochs starting from initial learning rate of 1e-4 which is reduced by $\times$10 at 10, 20 and 30 epochs, respectively.

SHP-Q8 optimized

We further optimize the previous model by incorporating classification loss. Leveraging the insight that deep learning classifiers extract features of different scales at different depths,

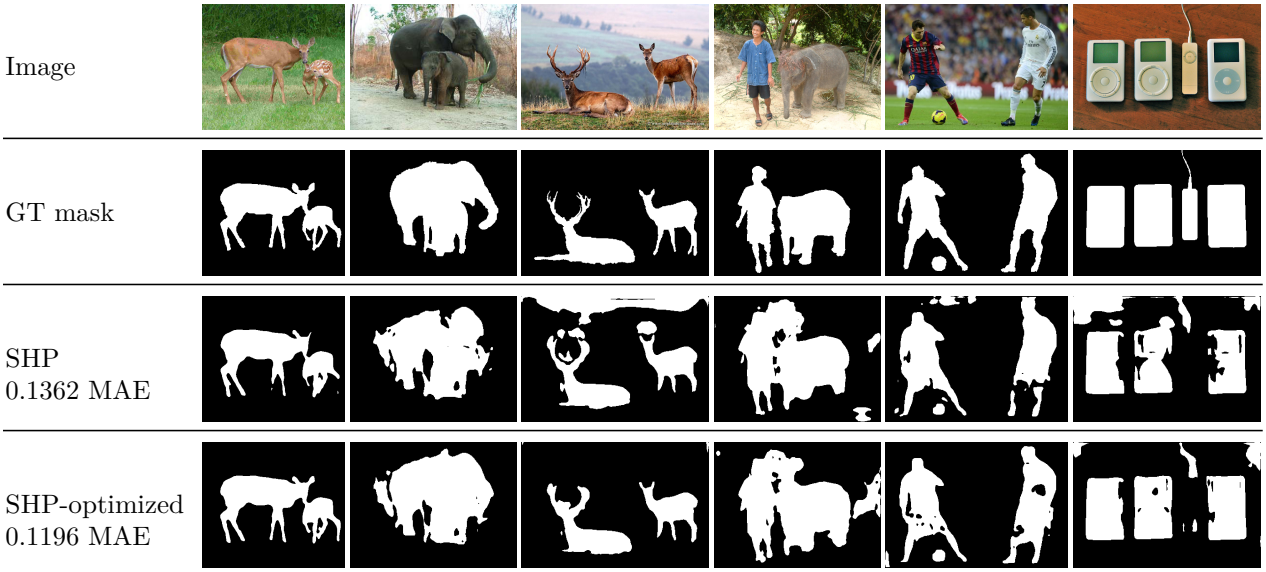


Figure 6.13: ROI masks generated by reusing the features extracted from the shared encoder backbone of SHP codec at $Q = 8$. When guided by classification, the proposed ROI prediction (row 4) improves by 12% in terms of Mean Absolute error (MAE) compared to the case without guiding (row 3).

we fine-tune the off-the-shelf SHP codec for $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance using the loss of Eq. (6.12) before training the saliency generator. By doing so, we anticipate that the encoder learns to acquire features important to both reconstruction and classification. During the early phase of fine-tuning SHP codec without saliency generation, we train the codec on ImageNet-1k data using Adam optimizer of learning rate 1e-3 for 1 epoch. Then we use the resulting model to initialize the SHP codec and continue the same steps used in SHP-$Q8$ to minimize saliency loss.

In Fig. 6.13, we compare the ROI predictions from these two settings against ground truths for HKU-IS data set. The ROI mask predictions from "SHP-$Q8$" and "SHP-$Q8$ optimized" are shown in row 3 and row 4. Note that both settings can generate ROI masks at sufficient accuracy. We observe 12% MAE reduction by optimizing SHP-$Q8$ with classification loss confirming the benefits of classification-guided saliency prediction.

6.7 CONCLUSIONS

This chapter presented a joint image compression and classification framework and its end-to-end optimization targeting ROI-driven applications. Our framework is consistent and amendable to $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance trade-off in resource constrained deployment. Targeting widespread deployment on low cost devices, we replaced the computationally-heavy feature extraction backbone for ROI prediction with a light-weight, feature-reusing encoder backbone guided by classification loss achieving faster, memory and power efficient inference. Based on extensive experiments on $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance over multiple data sets, our proposed joint image compression and classification framework demonstrates superior ROI rate-distortion and classification accuracy trade-off while maintaining sufficient overall image quality.

Part III

LDR-GUIDED CODECS

7

LINEAR DISCRIMINATIVE REPRESENTATION (LDR)-GUIDED LOW-SHOT LEARNING

As discussed in chapters 5-6, learning-based image/video codecs typically utilize the well known auto-encoder structure where the encoder transforms input data to a low-dimensional latent representation. "Efficient latent embedding" can improve DL task accuracy by preserving important features essential to the task and reduce bandwidth needs during compression for transmission/storage by removing redundancy of learned features. Finding such efficient latent embedding however is quite challenging considering the complexity of real-world-inputs and multiple post DL applications that rely on the learned latent embedding.

One natural approach to address this real-world data transformation problem is to learn latent representations that can be expressed using standard models or distributions. Along this direction, we explore the generalization of features of one such family of learned distributions called linear discriminative representation (LDR) [53] where latent embeddings are distributed in linear and class-wise orthogonal subspaces as introduced in Sec. 2.3. In particular, this chapter proposes a novel classification framework when training with extreme data imbalance scenarios named "low-shot training". We then evaluate the proposed framework for rate-accuracy performance focusing practical cloud-based networked classification applications. we limit our discussion in this chapter to two popular low-shot-learning instances, zero-shot learning (ZSL) and few-shot learning (FSL).

In ZSL and FSL, where training and inference are on disjoint classes, conventional cross-entropy (CE) based approaches tend to exhibit model over-fitting seen classes [132]. Hence, directly relying on seen class labels or attributes during training is disadvantageous. Recently proposed maximal coding-rate-reduction (MCR^2) principle finds LDRs with in-class compressive and inter-class discriminative properties by utilizing class labels as side information [53]. LDRs represent more universal features that are common to both seen and

unseen classes. This chapter proposes LDR-guided training to tackle challenging ZSL and FSL problems by introducing an extra feature embedding path. Tests over several benchmark data sets show that this LDR-guided models achieve significant accuracy improvement over conventional ZSL and FSL baselines with zero added memory or computational complexity during inference.

7.1 INTRODUCTION

In Zero-shot learning (ZSL), training data belong to a certain set of (seen) classes but the inference process needs to predict class labels of data (e.g., images) belonging to another set of (unseen) classes with no overlapping between seen and unseen class sets [50, 51, 133, 134]. A more practical setting, generalized zeros-shot learning (GZSL), aims at recognizing images from either seen or unseen classes during inference [54, 132]. Initial studies of ZSL have relied on abstract descriptions of image classes called "attributes" to handle classification problem of unseen classes [50, 54, 55]. Recent studies [132, 135, 136] have further expanded the use of "attributes" by learning better visual-semantic embeddings leading to state-of-the-art ZSL classification performance.

As a generalization of ZSL, few-shot learning (FSL) deals with cases when a few (N) labeled sample images are available for each class during inference [56–58]. The goal is to classify each sample image into one of the K classes. The set of N given labeled images for each class is the "support" set while the sample image for classification is the "query" image. This setup is commonly known as N-shot, K-way classification. In the same notation, ZSL with K unseen classes can be described as 0-shot, K-way classification. Similar to ZSL, learning universal (i.e., agnostic) feature embeddings that are essential in discriminating among classes, not limited to the seen set, remains as the core objective in FSL [137, 138].

When training and testing on disjoint classes, learning of visual-semantic embeddings in ZSL or feature embeddings in FSL with deep learning (DL) models relying directly on

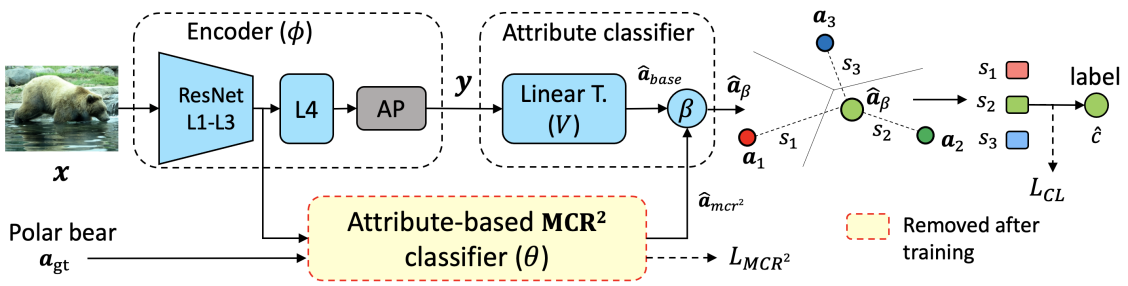


Figure 7.1: LDR-guided Zero-Shot Learning: we propose to use MCR^2-based loss during training to reduce over-fitting to seen classes by learning general features common in both seen and unseen classes. Zero extra computational cost is incurred during inference since MCR^2 classifier is removed after training. AP: average pooling.

conventional CE-based approaches can lead to severe over-fitting to the seen classes [132], because of practically unavoidable model over-parameterization. This leads to degradation of classification accuracy on unseen classes with more epochs trained on seen class images [139]. To alleviate this problem, recent works suggest an unseen validation set to select the least over-fitting model [136, 138]. Further, generative methods [140–142] that synthesize new unseen data during training have also found successes in ZSL. These approaches address the crucial need of learning visual-semantic or feature embeddings that are common for both seen and unseen classes towards better ZSL and FSL performance. Nevertheless, the practical utility of these approaches are limited by either the requirement for sufficient data or the complexity of data generation.

The authors of recent work [53] suggested the principle of maximal coding-rate-reduction (MCR^2) to promote linear representations $\boldsymbol{Z}$ of images $\boldsymbol{X}$ which are in-class compressive and inter-class discriminative by utilizing class labels as side information (See Fig. 2.6). Owing to the decoupled subspace structure of the learned representations $\boldsymbol{Z}$, the recent works [52, 59, 143] have further reported the success of MCR^2 in robust classification and reconstruction from learned features. In particular, the authors of [53] demonstrated model robustness to corrupted training labels. These works suggest the capacity of such learned LDRs to discriminate images to be less dependent on class labels. We advocate such feature to be favourable for low-shot learning tasks with highly imbalanced training data. Note that

optimization of MCR^2 objective introduces no extra parameters during inference and, hence, no complexity increase.

In this chapter, we address the challenging ZSL and FSL problems by introducing the use of LDR guidance. Our specific learning architecture, shown in Fig. 7.1, provides a new feature-embedding path for LDR-guided training. By promoting LDRs that are more general and less reliant on seen training classes, our LDR-guided models can learn features common to both seen and unseen classes and achieve significant accuracy improvement over conventional ZSL and FSL baselines. Furthermore, our LDR-guided ALE model achieves a new performance benchmark for ZSL on AwA2 data set, to the best of the authors' knowledge. Analytically, by considering cloud-based networked-AI application cases, we evaluate the rate-distortion and noise-robustness performance of the proposed LDR-guided ZSL models.

We organize the rest of the chapter as follows. In Section 7.2, we discuss common ZSL and FSL frameworks that serve as baselines in this analysis. Section 7.3 provides details of our proposed LDR-guided ZSL and FSL classification models, followed by experiments and test results in Sec. 7.4.

7.2 BACKGROUND AND RELATED WORKS

7.2.1 *ALE for Zero-Shot Learning*

In general, ZSL models showcase two disjoint transformations: from image domain to attribute domain and from attribute domain to class labels [50]. Since the training process only involves seen classes, learning a compatibility transformation between images and attributes becomes the core task [55]. The top part of Fig. 7.1 showcases the attribute-label-embedding (ALE) classifier [136, 144]. For each input image $\boldsymbol{x}$, the encoder $f_{\boldsymbol{\phi}}(\cdot)$ generates a low-dimensional latent vector $\boldsymbol{y} \in \mathbb{R}^d$, known as image embedding. Assuming k-dimensional ground truth

attributes, "attribute classifier" transforms $\boldsymbol{y}$ into attribute domain via a linear transformation $\boldsymbol{V} \in \mathbb{R}^{d\times k}$: $\hat{\boldsymbol{a}}_{\text{base}} = \boldsymbol{y}^\top \boldsymbol{V}$.

In this case, encoder and the attribute classifier jointly construct the compatibility function between images and attributes. During inference, corresponding class label c_j of K unseen classes is obtained by finding the ground truth class attribute $\boldsymbol{a}_{\text{gt}} \in \{\boldsymbol{a}_j\}_1^K$ that is closest to $\hat{\boldsymbol{a}}_{\text{base}}$ via inner-product.

$$c_j = \arg\max_{i\in[1,K]} \hat{\boldsymbol{a}}_{\text{base}} \cdot \boldsymbol{a}_i \tag{7.1}$$

In Fig. 7.1, we denote the similarity score by $s_i = \hat{\boldsymbol{a}}_{\text{base}} \cdot \boldsymbol{a}_i$.

7.2.2 *ProtoNets for Few-Shot Learning*

Fig. 7.2 illustrates the basic work flow of Prototypical Networks (ProtoNets) [137]. The encoder $f_{\boldsymbol{\phi}}(\cdot)$ transforms each image $\boldsymbol{x}$ in support set S to latent vectors $\boldsymbol{Z}_S = \{\boldsymbol{z}; \boldsymbol{z} = f_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{x} \in S\}$. For each class j in the support set, the average of latent vectors in that class $\boldsymbol{c}_j$ also lie in the same latent space. During inference, ProtoNet reuses the same encoder to obtain the

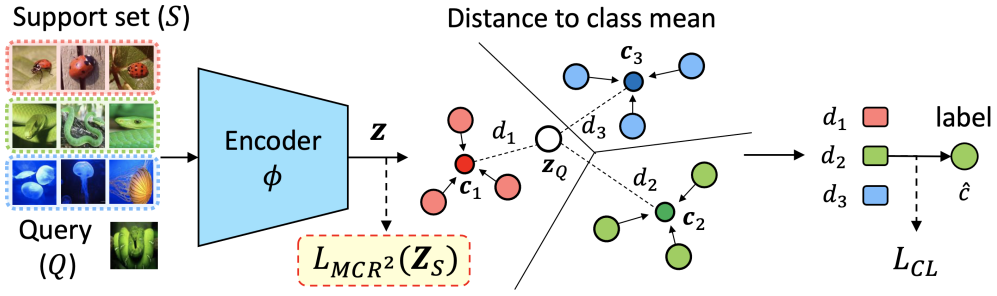


Figure 7.2: LDR-guided FSL model. We propose to map the images in support set S into linear discriminative representations $\boldsymbol{Z}_S$ by minimizing rate reduction loss L_{MCR^2}.

latent representation $\boldsymbol{z}_Q$ of the query image $\boldsymbol{x} \in Q$. Class label of $\boldsymbol{z}_Q$ is obtained by finding the class average $\boldsymbol{c} \in \{\boldsymbol{c}_j\}_1^K$ with the minimum distance $d(\cdot, \cdot)$ to $\boldsymbol{z}_Q$:

$$\hat{c} = \arg \min_{j \in [1,2,\cdots,K]} d(\boldsymbol{c}_j, \boldsymbol{z}_Q) \tag{7.2}$$

ℓ_2-norm has been the tested to be the best candidate to evaluate the distance in ProtoNet compared to other measures such as cosine-distance [137].

7.3 METHODS

We propose to guide the training of existing ZSL and FSL models with MCR^2 loss introduced in Sec. 2.3, to mitigate over-fitting to seen classes by learning general features that are universal among both seen and unseen classes.

7.3.1 *LDR-Guided Zero-Shot Learning*

Fig. 7.3 illustrates the proposed LDR-guided ZSL model. In addition to the attribute prediction $\boldsymbol{a}_{\text{base}}$ in the baseline, we introduce an additional attribute prediction $\boldsymbol{a}_{\text{mcr}^2}$ from an LDR $\boldsymbol{z}$.

Specifically, Fig. 7.3 shows a feature extraction backbone, in which each of ResNet [15] layers L1:L4 contains extracted features of different abstraction levels relevant to input image $\boldsymbol{x}$. Our proposed method reuses such extracted features to yield an LDR $\boldsymbol{z}$ by optimizing MCR^2 objective introduced in Sec. 2.3.1. The learned LDR $\boldsymbol{z}$ exhibits in-class compressive and inter-class discriminative characteristics based on attributes. In the following section, we describe our proposed approach of learning an attribute-based LDR $\boldsymbol{z}$.

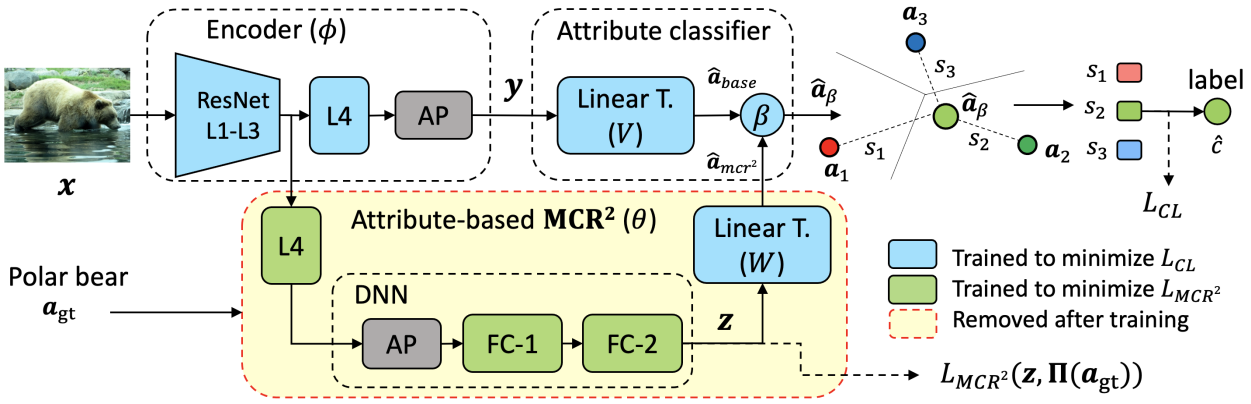


Figure 7.3: LDR-guided ZSL model. The proposed attribute based MCR^2 classifier generates an additional attribute prediction vector $\boldsymbol{a}_{\text{mcr}^2}$ based on LDR $\boldsymbol{z}$. AP: average pooling.

Attribute-Based Membership for MCR^2

Denote the set of latent vectors mapped from input image set $\boldsymbol{X}$ of a mini-batch $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_m]$, with batch size m, as $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_m] \in \mathbb{R}^{d \times m}$. We now formulate the corresponding attribute-based membership $\boldsymbol{\Pi}(A)$ based on ground truth attributes, instead of class labels proposed in [69]. In this case, membership set $\boldsymbol{\Pi}(A)$ consists of continuous diagonal matrices $\{\boldsymbol{\Pi}_1(A_1), \cdots, \boldsymbol{\Pi}_k(A_k)\} \in \mathbb{R}^{m \times m}$ with $\sum_{i=1}^{k} \boldsymbol{\Pi}_i(A_i) = \boldsymbol{I}_m$. Each $\boldsymbol{\Pi}_i(A_i)$ encodes the magnitude of attribute i for all samples in the mini-batch. We illustrate the proposed membership set construction in Fig. 7.4.

For instance, let there be two classes: polar bear and zebra, with a polar bear sample $\boldsymbol{x}_1$ and zebra sample $\boldsymbol{x}_2$. The given binary values for attributes {water, stripes, white} are $\{1, 0, 1\}$ and $\{0, 1, 1\}$ for polar bear and zebra classes, respectively; "1" indicates the presence of the corresponding attribute. We formulate the attribute-based membership using ℓ_1-normalized attribute vectors to ensure $\sum_{i=1}^{k} \boldsymbol{\Pi}_i(A_i) = \boldsymbol{I}_m$. With this, we write the attribute-based MCR^2 loss per mini-batch for a fixed distortion ϵ as

$$L_{\text{MCR}^2}(Z|\epsilon, \boldsymbol{\Pi}(A)) = -R(\boldsymbol{Z}|\epsilon) + R^c(\boldsymbol{Z}|\epsilon, \boldsymbol{\Pi}(A)). \tag{7.3}$$

End-to-End Loss

As shown in the lower half of Fig. 7.3, a ResNet layer (L4) and two fully connected layers (FC-1 and FC-2) are used to transform the intermediate features, extracted by the encoder, to the latent representation $\boldsymbol{z}$. Similar to the ALE classifier, the proposed model projects $\boldsymbol{z}$ into the attribute domain by using a linear transformation $\boldsymbol{W} \in \mathbb{R}^{d \times k}$: $\hat{\boldsymbol{a}}_{\text{mcr2}} = \boldsymbol{z}^\top \boldsymbol{W}$.

During training, we linearly combine the attribute embeddings $\boldsymbol{a}_{\text{base}}$ and $\hat{\boldsymbol{a}}_{\text{mcr2}}$ using a trainable vector $\boldsymbol{\beta}$ to generate a new attribute:

$$\boldsymbol{a}_\beta = \hat{\boldsymbol{a}}_{\text{base}} + \boldsymbol{\beta} \otimes \hat{\boldsymbol{a}}_{\text{mcr2}} \tag{7.4}$$

where $\otimes$ denotes the element-wise product. Similar to [136], we evaluate the classification loss of the proposed ALE classifier using the CE loss for image, ground truth attribute pair $\{\boldsymbol{x}, \boldsymbol{a}_{\text{gt}}\}$:

$$L_{\text{CL}}(\boldsymbol{x}, \boldsymbol{a}_{\text{gt}}) = -\log \frac{\exp(\boldsymbol{a}_\beta \cdot \boldsymbol{a}_{\text{gt}})}{\sum_1^K \exp(\boldsymbol{a}_\beta \cdot \boldsymbol{a}_i)} \tag{7.5}$$

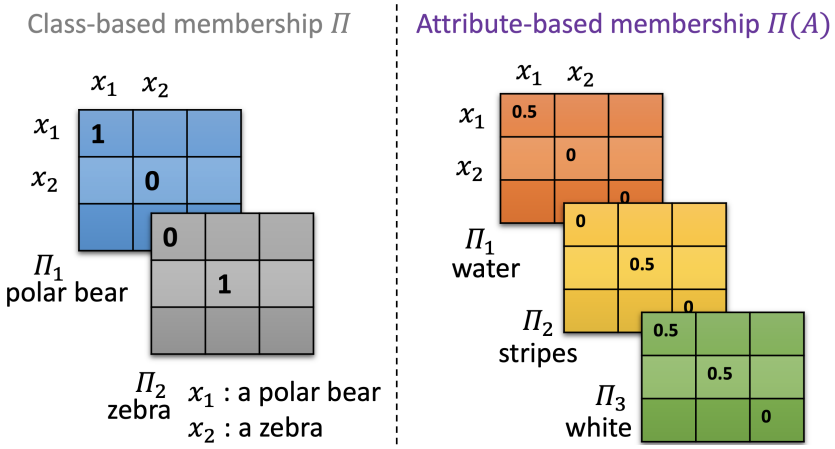


Figure 7.4: Examples of class vs attribute -based membership. We propose to encode the membership $\boldsymbol{\Pi}$ in MCR^2-based on normalized attributes to reduce over-fitting to seen classes.

We formulate the end-to-end loss function of the proposed LDR-guided model by combining Eqs. (7.3) and (7.5):

$$L(\boldsymbol{x}, \boldsymbol{a}_{\text{gt}}) = L_{\text{CL}}(\boldsymbol{x}, \boldsymbol{a}_{\text{gt}}) + \lambda L_{\text{MCR}^2}(Z|\epsilon, \boldsymbol{\Pi}(A)), \tag{7.6}$$

in which the hyper-parameter λ controls the trade-off between the classification loss and MCR^2 objective.

Disjoint Training

By minimizing Eq. (7.6), the end-to-end training finds the optimized parameter configurations of the encoder ($\boldsymbol{\phi}^*$), MCR^2 path ($\boldsymbol{\theta}^*$), linear transformations $\boldsymbol{V}^*$, $\boldsymbol{W}^*$ and $\boldsymbol{\beta}^*$,

$$\{\boldsymbol{\phi}^*, \boldsymbol{V}^*, \boldsymbol{\beta}^*, \boldsymbol{\theta}^*, \boldsymbol{W}^*\} = \arg \min_{\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{\beta}} \sum_{\{x, a_{\text{gt}}\}} L(\boldsymbol{x}, \boldsymbol{a}_{gt}) \tag{7.7}$$

In order to ensure that ALE-based embedding $\hat{\boldsymbol{a}}_{\text{base}}$ and MCR^2-based embedding $\hat{\boldsymbol{a}}_{\text{mcr2}}$ are mutually beneficial with exclusive features, we propose the following disjoint training strategy. We only update the parameters $\boldsymbol{\phi}, \boldsymbol{V}, \boldsymbol{\beta}$ (marked in blue) in Fig. 7.3 based on L_{cls} and only update $\boldsymbol{\theta}$ and $\boldsymbol{W}$ (marked in green) based on L_{MCR^2}. This disjoint process further facilitates the standalone operation of the encoder and the attribute classifier during inference without the MCR^2 path. Furthermore, it helps guide ZSL to find more universal features that are common to both seen and unseen classes. Such process is less likely to converge to over-fit models. Once training is complete, we remove the MCR^2 path in yellow dashed block in Fig. 7.3 and bring back the model to its former computational complexity.

7.3.2 *LDR-Guided Few-Shot learning*

We illustrate the proposed LDR-guided FSL model in Fig. 7.2. Similar to LDR-guided ZSL, we promote latent feature embedding $\boldsymbol{z}$ of the FSL encoder to be in-class compressive and inter-class discriminative with MCR^2 loss. By doing so, we expect to introduce two benefits to improve FSL classification. In-class compressiveness helps each latent embedding of class j of the support set to be closer to their class mean $\boldsymbol{c}_j$. Inter-class discirminability pushes the class averages $\boldsymbol{c}_j$ further apart from each other. We note that recent works [138, 145, 146] have proposed "embedding adaptation" functions to induce similar effects.

In this work, we only use the support set $\boldsymbol{X}_S = \{\boldsymbol{x} \in S\}$ during training to minimize the MCR^2 loss although, support images with class labels are available during inference in contrast to ZSL. For a given encoder instance $f(\cdot, \boldsymbol{\phi})$ and latent vectors of the support set $\boldsymbol{Z}_S(\boldsymbol{\phi})$ and query $\boldsymbol{z}_Q(\boldsymbol{\phi})$, we write the MCR^2 loss function for fixed ϵ similar to Eq. (7.3):

$$L_{\text{MCR}^2}(\boldsymbol{Z}_S|\epsilon, \boldsymbol{\Pi}) = -R(\boldsymbol{Z}_S|\epsilon) + R^c(\boldsymbol{Z}_S|\epsilon, \boldsymbol{\Pi}). \tag{7.8}$$

Since the ground truth class labels are available for the support set, we rely on class-based membership when formulating $\boldsymbol{\Pi}$. Classification loss L_{cls} in the proposed LDR-guided FSL model measures the softmax over distance between $\boldsymbol{z}_Q$ and its corresponding class center $\boldsymbol{c}_{\text{gt}}$ in the feature embedding space [137]. For a training image-label instance $\{\boldsymbol{x}_Q, c_{\text{gt}}|\boldsymbol{z}_Q = f_{\boldsymbol{\phi}}(\boldsymbol{x}_Q), \boldsymbol{x}_Q \in Q\}$, we write:

$$L_{\text{CL}}(\boldsymbol{x}_Q, c_{\text{gt}}|\boldsymbol{X}_S) = -\log \frac{\exp(-d(\boldsymbol{z}_Q, \boldsymbol{c}_{\text{gt}}))}{\sum_{j=1}^{K} \exp(-d(\boldsymbol{z}_Q, \boldsymbol{c}_j))}. \tag{7.9}$$

Similar to LDR-guided ZSL case, we formulate the end-to-end loss function for the FSL case as a weighted sum of classification and MCR^2 loss.

$$L(\boldsymbol{x}_Q, c_{\text{gt}}|\boldsymbol{X}_S) = L_{\text{CL}}(\boldsymbol{x}_Q, c_{\text{gt}}|\boldsymbol{X}_S) + \lambda L_{\text{MCR}^2}(\boldsymbol{Z}_S|\epsilon, \boldsymbol{\Pi}). \tag{7.10}$$

Again, the hyper-parameter λ leverages trade-off between the classification loss and MCR^2 loss. Since no additional parameters are dedicated for the MCR^2 branch, we optimize the encoder parameters based on both above loss terms to find the encoder configuration $\boldsymbol{\phi}^*$:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \sum_{\{\boldsymbol{x}_Q, c_{\text{gt}}, \boldsymbol{X}_S\}} L(\boldsymbol{x}_Q, c_{\text{gt}}|\boldsymbol{X}_S) \tag{7.11}$$

7.4 EXPERIMENTS AND RESULTS

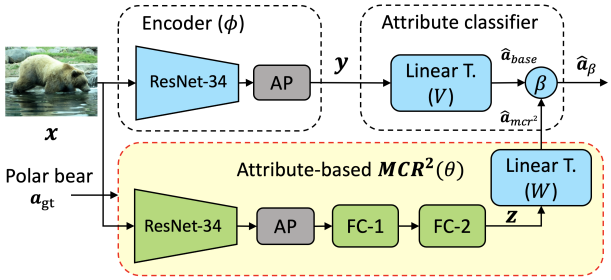
We now provide the details of the experiment setup and the results of MCR^2-guided models in comparison to state-of-the-art (SOTA) benchmarks and models.

7.4.1 *LDR-Guided Zero-Shot Learning*

We conducted ZSL experiments on three common ZSL benchmark data sets: AwA2, CUB, and SUN. AwA2 [147] contains 37,322 images of 50 animal classes with 85 attributes per class. We used the standard training and test (40 seen/10 unseen) split proposed in [54]. CUB [148] contains 11,788 bird images from 150 seen and 50 unseen classes with 312 attributes per class. Similarly, SUN [149] contains 14,340 scene images from 645 seen and 72 unseen classes, with 102 attributes per class.

Method	ZSL	GZSL		
	T1	u	s	H
ALE (baseline)	56.2	52.8	60.8	56.5
LDR-guided ALE-a	55.9	52.4	62.3	56.9
LDR-guided ALE-b	**57.1**	**53.2**	**64.0**	**58.1**

(a) Results



(b) LDR-guided ALE-b model

Figure 7.5: Ablation study: ZSL and GZSL accuracy comparison on AwA2 data set with ResNet-34 feature extraction backbone

Baseline ALE Classifier

We used the ALE classifier described in Sec. 7.2.1 as the baseline for ZSL. Applying the implementation for the "base mode classifier" given in [136], we used ResNet-101, pre-trained on ImageNet-1k [77], as feature extraction backbone. We pre-train the classifier parameters $\boldsymbol{V}$ on seen class images for 2 epochs to minimize L_{CL} followed by optimizing the full network end-to-end for another 28 epochs at reduced learning rate that is $\times 0.001$ smaller.

We evaluate the baseline ZSL and GZSL task accuracy using the same model trained as described above. For GZSL evaluation, we used the calibrated stacking [139] parameter values recommended in [136].

LDR-Guided ALE Classifier

We initialized the encoder backbone with the same pre-trained ResNet-101 as in baseline case. Further, we initialized the green L4 layer of MCR^2 path, as shown in Fig. 7.3, using the same pre-trained parameter values of the blue L4 layer in the encoder backbone.

We pre-train the classifier parameters $\boldsymbol{V}$, linear transformations W and $\boldsymbol{\beta}$ on seen class images for 2 epochs. Then during each subsequent epoch, we apply the disjoint training approach proposed in Sec. 7.3.1 as follows. First, we update parameters $\boldsymbol{\phi}$, $\boldsymbol{V}$ and $\boldsymbol{\beta}$ to minimize the CE loss. We then optimize the MCR^2 path parameters $\boldsymbol{\theta}$ on MCR^2 loss. We

Table 7.1: ZSL and GZSL accuracy comparison. 'LFGAA+H' indicates LFGAA+Hybrid [1] method.

	Zero Shot			Generalized Zero Shot								
	AwA2	CUB	SUN	AwA2			CUB			SUN		
Method	T1	T1	T1	u	s	H	u	s	H	u	s	H
AREN [150]	67.9	71.8	60.6	15.6	92.9	26.7	38.9	78.7	52.1	19.0	38.8	25.5
LFGAA+H	68.1	67.6	61.5	27.0	93.4	41.9	36.2	80.9	50.0	18.5	40.4	25.3
APN [136]	**68.4**	**72.0**	**61.6**	56.5	78.0	65.5	65.3	69.3	**67.2**	**41.9**	34.0	**37.6**
ALE (baseline)	66.8	69.9	60.6	55.3	81.2	65.8	64.6	66.4	65.5	37.9	34.9	36.3
LDR-guided ALE	67.7	70.4	60.8	**58.5**	78.7	**67.2**	**65.5**	67.1	66.3	38.1	35.2	36.6

repeat the two alternating steps for 58 more epochs and record the final ZSL and GZSL test accuracy. Same calibrated stacking parameter values in [136] are used for GZSL.

Ablation Study

In this experiment, we study the effect of feature sharing for the MCR^2 path. We use a ResNet-34 backbone for both the baseline and LDR-guided models and follow the same training process in Sec. 7.4.1.

LDR-guided ALE-a model has the same architecture as Fig. 7.3 but with a ResNet-34 backbone. LDR-guided ALE-b model as shown in Fig. 7.5(b) learns features from raw RGB pixel values, enabling higher flexibility for MCR^2 path to generate exclusive features. Although this introduces additional computationally cost during training, the benefit is the significant improvement of both ZSL and GZSL accuracy. See Table. 7.5(a). Furthermore, in the inference phase, there is no increase in computational complexity when compared with the baseline ALE since the MCR^2 path will be removed.

SOTA Comparison

Table. 7.1 provides the comparative ZSL and GZSL accuracy of the proposed LDR-guided ALE approach with existing ZSL methods on AwA2, CUB, and SUN data sets. We observe that the LDR-guided ALE exhibit consistently superior performance in terms of top-1 ZSL accuracy (T1), unseen (u) accuracy, and H-mean (H) accuracy values of GZSL. In particular,

our proposed method reports 3.2% of unseen accuracy and 1.4% total accuracy gains over ALE baseline for AwA2 data set. In comparison with the GZSL accuracy reported in [136], Table. 7.1 shows accuracy gains of 2% in unseen and 1.7% in terms of H-mean for our proposed LDR-guided ALE. To the best of our knowledge, these results record a new state-of-the-art benchmark for AwA2 data set under GZSL category.

It is noteworthy that LDR-guided training exhibits clearer improvements for GZSL category compared to ZSL. This result further suggests that the MCR^2 principle tends to induce more universal features that are common to both seen and unseen classes. Please refer to Appendix E for detailed explanations of model architectures, additional ablation tests and robustness comparisons of the proposed LDR-guided models.

7.4.2 *LDR-Guided Few-Shot Learning*

For FSL experiments, we relied on standard FSL benchmark data set: mini-ImageNet [58], including a total number of 100 classes and 600 images per class. We adopted the same setup of [151], and used 64 classes as seen, 16 and 20 as two sets of unseen categories for model validation and testing, respectively.

FSL Classifiers

We experimented with four different encoder models : Conv-4 [137], ConvNet, ResNet-10 [15] and ResNet-12. Using ProtoNets [137] as the main baseline model, we trained and tested the classifier with the aforementioned encoder backbones for 1-shot and 5-shot classification. For each encoder-shot combination, we trained a corresponding LDR-guided model for comparison.

We train both baseline and LDR-guided models for 200 epochs and select the best model parameters based on the standard validation set as the final model. Following [138], the final model was evaluated on 10,000 random FSL test cases.

Ablation Study

This experimental setup aims to examine the capability of LDR-guided models to extract common features for classes seen during training and the few shots seen during inference. For this purpose, we combine standard training and validation sets of mini-ImageNet to generate a new training set. We test this hypothesis by training the baseline and LDR-guided models for 200 epochs without using a validation set to select the best model intentionally to highlight the impact of over-fitting. After training, we evaluate the final model on the standard mini-ImageNet test set. Models capable of extracting universal features are expected to record high test accuracy without apparent over-fitting to the classes seen during training.

We experimented on two encoder backbones: Conv-4 and ResNet-10. To check the generalization of the models, we trained the models with and without image augmentation. 'ProtoNet-a' and 'LDR-guided ProtoNet-a' use the common FSL image augmentations of random horizontal flips and color jitters following [137]. Table. 7.2 records the 5-way and 10-way results on mini-ImageNet test set.

Table 7.2: Ablation study: FSL accuracy comparison on mini-ImageNet data set

		5-way		10-way	
Method	Encoder	1-shot	5-shot	1-shot	5-shot
ProtoNet [137]	Conv-4	42.6	63.2	33.0	48.2
LDR-guided ProtoNet	Conv-4	46.8	64.4	33.5	49.6
ProtoNet	ResNet-10	47.4	64.4	32.7	48.6
LDR-guided ProtoNet	ResNet-10	53.8	69.8	38.0	54.1
ProtoNet-a	ResNet-10	53.0	71.4	36.8	55.5
LDR-guided ProtoNet-a	ResNet-10	**56.4**	**74.4**	**38.0**	**58.4**

We observe that the LDR-guided models demonstrate consistent accuracy improvement over corresponding baseline models for each of these considered cases. In particular, with ResNet-10 encoder backbones, the proposed LDR-guided ProtoNet exhibits absolute accuracy improvement of 5.3%-6.4% under no-augmentation case in comparison to the baseline. Similar to baseline models, LDR-guided models also significantly benefit from image augmentation

during training. Interestingly, the improved accuracy of the baseline models owing to image augmentation closely match the accuracy of LDR-guided models without augmentation. This observation further suggests the strength of the proposed models in mitigating the problem of over-fitting on classes seen during training.

Comparison with Recent FSL Methods

In this study, we follow the implementation in [138] to retrain the recent FSL embedding adaptation approaches such as FEAT, GCN, Bi-LSTM, and DeepSet by integrating the LDR-guided training approach. This integration leads to LDR-guided FEAT, LDR-guided GCN, LDR-guided Bi-LSTM, and LDR-guided DeepSet, respectively. Table. 7.3 shows the 5-way accuracy. Similar to LDR-guided ProtoNets, we minimized the MCR^2 loss on the support set in other LDR-guided models. Both baseline and LDR-guided models used pre-trained encoder initialization provided in [138].

Table 7.3: LDR-guided FSL 5-way classification accuracy comparison with SOTA methods on mini-ImageNet data set. For each existing method compared, we provide the performance its of LDR-guided version

	ConvNet		ResNet-12	
Method	1-shot	5-shot	1-shot	5-shot
MatchNet [58]	48.14	-	62.63	-
LDR-guided MatchNet	48.89	-	62.82	-
ProtoNet [137]	51.61	69.56	62.39	78.17
LDR-guided ProtoNet	51.63	69.75	63.89	78.49
GCN [138]	52.13	69.48	62.07	77.38
LDR-guided GCN	52.31	69.67	63.28	77.58
Bi-LSTM [138]	48.64	69.70	62.81	77.30
LDR-guided Bi-LSTM	48.75	69.78	63.72	78.05
DeepSet [138]	50.88	68.41	62.02	78.00
LDR-guided DeepSet	51.20	68.45	62.34	78.01
FEAT [138]	53.38	70.21	65.31	80.84
LDR-guided FEAT	**53.47**	**70.28**	**65.74**	**80.94**

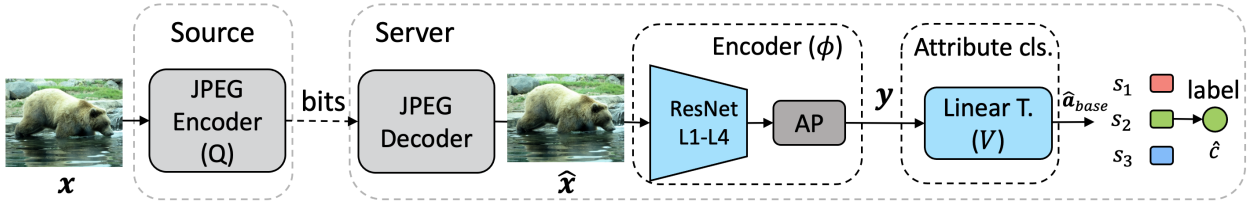
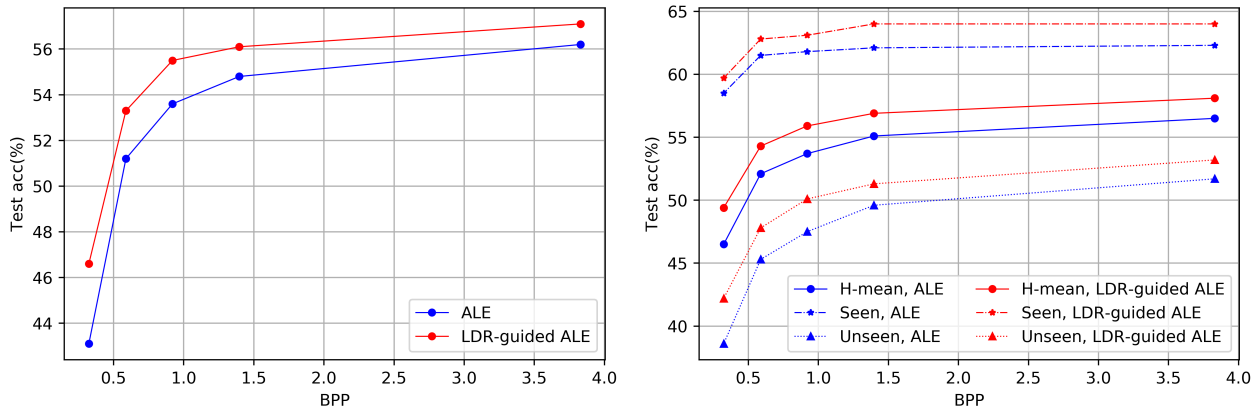


Figure 7.6: Conventional cloud-based ZSL on JPEG compressed images. Embedded source device encodes image $\boldsymbol{x}$ into a bit-steam using JPEG encoder and transmits to a remote server to infer the class label $\hat{c}$.

7.4.3 $\mathcal{R}$-$\mathcal{C}$ *Performance on JPEG-Compressed Images*

Considering future deployment of edge and cloud networked learning under limited channel bandwidth, we aim to evaluate the rate-classification performance of ZSL tasks in a networked inference setting of Fig. 7.6. A source device with limited computational power compresses input image $\boldsymbol{x}$ to a bit stream with a standard JPEG [62] encoder with quality parameter $q \in [1, 100]$. During cloud inference, the resource-rich server reconstructs the image $\hat{\boldsymbol{x}}$ from encoded data bits transmitted by the source device using a standard JPEG decoder.

We benchmark the rate-accuracy performance of the considered baseline ALE classifier and the proposed LDR-guided ALE classifier on JPEG compressed images of AwA2 data set. For each quality value q, we retrain the models with training images compressed at q and



(a) ZSL (b) GZSL

Figure 7.7: Rate-accuracy performance for AwA2 data set under the proposed LDR-guided framework. Rate is recorded in bits per pixel (BPP). All considered models consist of a ResNet-34 feature extraction backbone.

evaluate the performance on the test set compressed at the same q. Rate was measured in bits per pixel (bpp). Fig. 7.7 shows the natural classification accuracy loss in every considered case when the data rate of the network link drops, following the general rate-classification characteristics [152, 153]. Importantly, the LDR-guided classifier outperforms the baseline ALE classifier over the entire tested data rate range. For instance, we observe 3.5% and 2.9% accuracy gains for ZSL and GZSL, respectively, at 0.324 bpp. Furthermore, the accuracy gain of the LDR-guided approach over the baseline consistently improves at lower data rates under both ZSL and GZSL scenarios.

Interestingly, Fig. 7.7(b) shows that the accuracy gain of LDR-guided models for the seen class gradually dissipates with increasing accuracy gains for the unseen class at low bpp. This observation suggests that more priority is given to learning the universal features at lower bandwidths by the LDR-guided models to benefit unseen classes.

7.5 CONCLUSIONS

In this chapter, we introduced a new classification framework for ZSL and FSL tasks. Specifically for ZSL and FSL, we leverage the MCR^2 principle to mitigate the possibility of over-fitting to seen classes observed in conventional CE-based approaches when training and testing on disjoint classes. Relying on the ground truth class labels/attributes only as side information, the proposed LDR-guided classifiers learn more universal features for both seen and unseen classes. Tested over several standard data sets, our LDR-guided models exhibit significant accuracy improvement over conventional ZSL and FSL baselines without additional memory or computational complexity during inference. Maintaining computational competitiveness with SOTA methods, our LDR-guided classifier model achieves a new benchmark for GZSL on AwA2 data set. Next chapter discusses a joint compression and classification framework based on LDRs.

8

LDR-BASED HIERARCHICAL GROUPING FOR JOINT IMAGE COMPRESSION AND CLASSIFICATION

Previous chapter investigated the applications of LDR in data imbalanced scenarios and briefly demonstrated rate-accuracy performance of JPEG compressed images on LDR-guided models. However, the proposed LDR-guided classification in Chapter 7 is most likely suboptimal since it does not harvest the compressibility of the learned LDR and its end-to-end optimization for the joint compression and classification. Considering cloud-based image classification similar to Sec. 7.4.3, we explore the compression of learned LDR for improved rate-accuracy performance.

In particular, this chapter examines the effect of assigning high level coarse grouping labels to each learned latent subspace. Designing coding profiles for each latent group can achieve high compression during encoding. We show that such grouping can be learned via end-to-end optimization of the codec and DL model to optimize rate-accuracy for a given data set. For cloud-based inference, source encoder can select a coding profile based on its learned grouping and encode the data features accordingly. Our test results on image classification show that significant performance improvement can be achieved with learned grouping over its non-grouping counterpart.

8.1 INTRODUCTION

Deep learning applications on images and video data generated by distributed low end devices are continuously expanding at a staggering pace. In such networked AI scenarios, low end devices such as roadside cameras, vehicle sensors, and IoT devices are in charge of data capturing before transporting them to cloud/edge servers with high memory and

computational capacity required for executing machine tasks. Under limited (wireless) network bandwidth, image/video data must be compressively encoded for transport channels without sacrificing ML task accuracy or visualization quality at the remote end [31, 78, 104].

Another important consideration in cloud/edge-based deep learning is the reliability of training data. Inference performance of supervised learning tasks such as image classification, object recognition, and segmentation, depends critically on the accuracy of labeled data available for training [154]. Training samples can be mislabelled due to human errors and occasional corruptions during transmission and storage. Hence, an equally important problem in distributed ML is to learning algorithms, robust to training mislabels [155].

Recently developed image/video codecs based on deep learning often feature an auto-encoder structure [33, 34, 67]. The encoder maps the high dimensional ($\mathbb{R}^D$) input manifold/space of high complexity to a low dimensional ($\mathbb{R}^d$) latent representation. Key features of the latent representation are acquired from end-to-end optimization including the codec and media processing model. In this chapter, we aim to addresses both aforementioned problems and optimizes the latent representation for efficient encoding and effective ML.

We organize this chapter as follows. In Section 8.2, we revisit the Maximal Coding Rate Reduction (MCR^2) principle of finding an LDR in view of data compression. We build a direct connection of MCR^2 with efficient latent encoding for transmission in the distributed deep learning paradigm. We then propose an end-to-end optimization of networked image classification system by leveraging the concept of data grouping in MCR^2 in Sec. 8.3. Finally, we present test results on CIFAR-100 and ImageNet-1k (size 32) data sets in Sec. 8.4.

8.2 BACKGROUND AND RELATED WORKS

Following the same notation used in the previous chapter, let $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_1}, \cdots \boldsymbol{x_m}] \in \mathbb{R}^{D\times m}$ be m i.i.d. samples of dimension D. An encoder $f(\cdot, \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ maps each

sample $\boldsymbol{x}$ to a d-dimensional ($d < D$) learned representation $\boldsymbol{z}$ such that $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{\theta})$. We write the set of latent vectors mapped from set $\boldsymbol{X}$ as $\boldsymbol{Z} = [\boldsymbol{z_1}, \boldsymbol{z_2}, \cdots, \boldsymbol{z_m}] \in \mathbb{R}^{d \times m}$.

$$\boldsymbol{x} \xrightarrow[f(\cdot,\boldsymbol{\theta})]{\text{Encoder}} \boldsymbol{z}(\boldsymbol{\theta}) \xrightarrow{\text{Channel}} \hat{\boldsymbol{z}}(\boldsymbol{\theta}) \xrightarrow[g(.,\boldsymbol{\phi})]{\text{Classifier}} y \tag{8.1}$$

Source device uses entropy coding to encode $\boldsymbol{z}$ for transmission. The receiver decodes the code words to obtain reconstructed $\hat{\boldsymbol{z}}$. An ML task model $g(\cdot, \boldsymbol{\phi})$ with parameters $\boldsymbol{\phi}$ generates the output label $y = g(\hat{\boldsymbol{z}}, \boldsymbol{\phi})$ based on $\hat{\boldsymbol{z}}$.

8.2.1 *Non-asymptotic rate-distortion with multiple groups*

The authors of [69] provided a tight upper bound on the number of bits required to encode $\boldsymbol{X}$ in a subspace. For a Gaussian channel with distortion ϵ^2, defined as the ℓ_2-norm of reconstruction error, the mean code length per sample (for large m) is [69]:

$$R(\boldsymbol{Z}|\epsilon) = \frac{1}{2} \log_2 \det \left[\boldsymbol{I} + \frac{d}{m\epsilon^2} \boldsymbol{Z}\boldsymbol{Z}^\top \right] \text{ bits.} \tag{8.2}$$

This result requires $\boldsymbol{Z}$ to be within the same d-dim subspace. Partition $\boldsymbol{Z}$ into k disjoint subsets (groups) based on features: $\boldsymbol{Z} = \boldsymbol{Z}_1 \cup \cdots \cup \boldsymbol{Z}_k$. The grouping of $\boldsymbol{Z}$ can be denoted by a membership set $\boldsymbol{\Pi}$ of binary diagonal matrices $\{\boldsymbol{\Pi}_1, \cdots, \boldsymbol{\Pi}_k\} \in \mathbb{R}^{m \times m}$ with $\sum_{i=1}^{k} \boldsymbol{\Pi}_i = \boldsymbol{I}_m$. Each group has size $\operatorname{tr}(\boldsymbol{\Pi}_j)$. For this case, [69] similarly provided a tight upper bound of average rate (bits per sample) when $m \gg d$:

$$R^c(\boldsymbol{Z}|\epsilon, \boldsymbol{\Pi}) = \sum_{j=1}^{k} \frac{\operatorname{tr}(\boldsymbol{\Pi}_j)}{2m} \log_2 \det \left[\boldsymbol{I} + \frac{d \cdot \boldsymbol{Z}\boldsymbol{\Pi}_j\boldsymbol{Z}^\top}{\operatorname{tr}(\boldsymbol{\Pi}_j)\epsilon^2} \right] \tag{8.3}$$

8.2.2 *Maximal coding for rate reduction (MCR^2)*

To find an LDR $\boldsymbol{Z}(\boldsymbol{\theta}) = f(\boldsymbol{X}, \boldsymbol{\theta})$, the MCR^2 principle of [53] maximizes the loss function of

$$\Delta R(\boldsymbol{Z}(\boldsymbol{\theta})|\epsilon, \boldsymbol{\Pi}) = R(\boldsymbol{Z}(\boldsymbol{\theta})|\epsilon) - R^c(\boldsymbol{Z}(\boldsymbol{\theta})|\epsilon, \boldsymbol{\Pi}). \tag{8.4}$$

As shown in [53], the first term of (8.4) measures the code length for all features in $\boldsymbol{Z}$ whereas the second term is the sum code length of features in each of the k groups. Treating $\boldsymbol{z}$ as the output of the final feature layer in a ResNet [15] classifier, the authors of [53] used a simple subspace classifier after applying true class labels to generate the membership set $\boldsymbol{\Pi}$. This classifier on the learned LDR is robust against mislabeled images during training.

8.2.3 *MCR^2 and Latent Encoding*

Cloud-based deep learning (DL) applications involve low-end devices to capture images and videos for encoding and transmitting to powerful computing nodes to carry out learning. To reduce power and bandwidth consumption, it is vital to efficiently encode the latent representation $\boldsymbol{Z}$ by minimizing $R^c(\boldsymbol{Z}(\boldsymbol{\theta})|\epsilon, \boldsymbol{\Pi})$ for transmission. Instead of only minimizing $\Delta R(\boldsymbol{Z}(\boldsymbol{\theta})|\epsilon, \boldsymbol{\Pi})$ as in MCR^2, the need to improve code efficiency by constraining $R^c(\boldsymbol{Z}(\boldsymbol{\theta})|\epsilon, \boldsymbol{\Pi})$ is practically significant. In fact, maximizing the MCR^2 objective in Eq. (8.4) does not necessarily guarantee to reduce $R^c(\boldsymbol{Z}(\boldsymbol{\theta})|\epsilon, \boldsymbol{\Pi})$. In this chapter, we propose a framework with the dual objective of achieving robustness against mislabeling and reducing the rate $R^c(\boldsymbol{Z}(\boldsymbol{\theta})|\epsilon, \boldsymbol{\Pi})$ of efficient latent encoding by leveraging the grouping information.

8.2.4 *Latent Compression in View of Grouping*

Several previous works utilized grouping for better rate-distortion performance in image/video compression [156–158]. The authors of [156] proposed dividing videos into clusters, each with distinct encoding profile in video compression for transmission. For image compression, the authors of [158] proposed to encode highly correlated images together to improve overall compression ratio. Similarly, we explore the benefits of the grouping to extract compressive features for a given DL task under bandwidth constraint.

Optimizing image/video compression codecs for a given task is an active area of research. Some recent works [31, 78, 104] showed promises by jointly optimizing the codec and DL model including the entropy coding parameters. Since the learned representation $\boldsymbol{Z}$ is not unique, the encoder $\boldsymbol{\theta}$ and task model $\boldsymbol{\phi}$ can be end-to-end optimized for a learning objective such as classification or segmentation.

Grouping information helps generate membership set $\mathbf{\Pi}$. Candidates for grouping information include ground truth labels, coarse labels and tags (metadata), depending often on the learning task. For instance, in hierarchical classification, coarse labels have shown benefits for improving classification accuracy [159, 160]. Similarly, Metadata is less prone to error compared to ground truth labels since accurate manual labelling is not needed.

In practice, initial grouping information may not be available at acquisition or may not be a good candidate for a given task. To be broadly accommodating, we propose that the grouping be learned through unsupervised learning for end-to-end optimization to achieve better $\mathcal{R}$-$\mathcal{C}$ trade-off. In real time applications, source device can then select a pre-trained coding profile per group based on the $\mathcal{R}$-$\mathcal{C}$ trade-off.

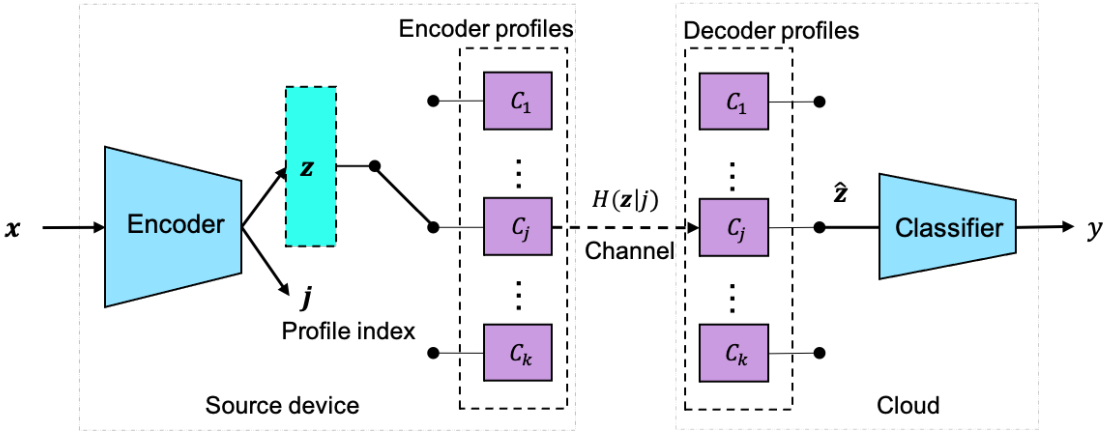
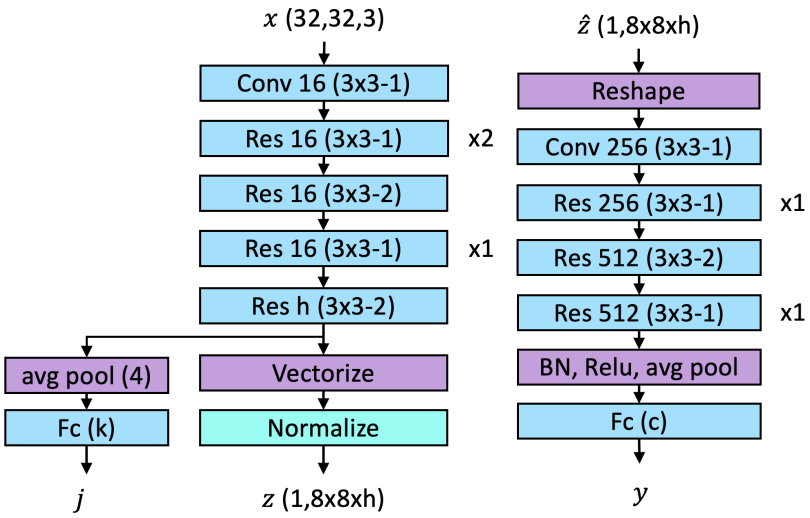


Figure 8.1: Proposed framework. During inference, for input image $\boldsymbol{x}$, the encoder generates a profile index j and the feature vector $\boldsymbol{z}$. Then, the encoder profile of C_j encodes $\boldsymbol{z}$ to a bit stream with entropy $\mathrm{H}(\boldsymbol{z}|j)$.

8.3 PROPOSED END-TO-END FRAMEWORK

Our proposed framework in Fig. 8.1 consists of an encoder, a set of coding profiles, and a task model (e.g. an image classifier). Encoder maps image $\boldsymbol{x}$ to a low-order latent vector $\boldsymbol{z}$ and generates a group label $j \in \{1, 2, \cdots, k\}$, referred to as the "profile index". This profile set has k different encoding-decoding profiles $\{C_j\}_1^k$, each of which is optimized to compress latents with profile index j. For image encoding, a profile may typically consist of a quantizer, an entropy coder, an entropy decoder and an optional de-quantizer. Classifier uses the decoded latent $\hat{\boldsymbol{z}}$ as input for classification into c classes with $c \geq k$.

During training of the end-to-end architecture, parameters of the encoder, coding profile, and the classifier are jointly optimized. The optimized encoder and decoder profiles are stored at the source and the cloud nodes, respectively. During inference phase, the source encoder determines the profile index j for the input image $\boldsymbol{x}$ and generates the feature vector $\boldsymbol{z}$. Next, the encoder uses profile C_j to encode $\boldsymbol{z}$ into bit stream for transmission to the classifier node on cloud/edge. The classifier decodes the received bit stream based on profile C_j to recover the feature vector $\hat{\boldsymbol{z}}$ for subsequent classification.



(a) Encoder (b) Classifier

Figure 8.2: Encoder and classifier architectures used for CIFAR-100 ($c = 100$) and ImageNet-1k ($c = 1000$). Number of filters of the last ResNet block of the encoder are $h = 16, 12$ for hidden sizes $d = 1024, 768$, respectively. "Conv 16 (3x3-2)" represents a 2D convolution block with 16 filters of size 3×3 and stride of 2. "Res 16 (3x3-2)" represents a basic ResNet block [15] with down-sampling factor 2.

8.3.1 *Rate under Quantization Noise*

Quantization is the primary source of rate reduction in both commercial [97] and recent DL based image/video codes [33, 67]. Hence, we model the wireless channel with quantization noise $\boldsymbol{n} \in \mathbb{R}^d$ as follows. Let $\boldsymbol{z} \in \mathbb{R}^d$ be the recovered vector from the quantized vector $\hat{\boldsymbol{z}} \in \mathbb{R}^d$. Then it is clear that $\hat{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{n}$. Following previous works [33, 104], for quantization step s, elements of $\boldsymbol{n}$ can be modeled as independent, zero mean, and uniformly distributed in $[-s/\sqrt{d}, s/\sqrt{d}]$. Hence, the distortion ϵ from quantization noise forms an upper bound of reconstruction error:

$$\mathbb{E}[||\boldsymbol{z} - \hat{\boldsymbol{z}}||_2] \leq \epsilon^2 = \frac{(2s)^2}{12} \tag{8.5}$$

Following [69], we can derive an upper bound for the rate at a given distortion ϵ using the "sphere packing" principle [161] in information theory. To begin, estimate covariance matrix

$$\hat{\boldsymbol{\Sigma}} = \mathbb{E}[\frac{1}{m}\sum_{i=1}^{m}\hat{\boldsymbol{z}}_i\hat{\boldsymbol{z}}_i^\top] = \frac{1}{m}\boldsymbol{Z}\boldsymbol{Z}^\top + \frac{\epsilon^2}{d}\boldsymbol{I}_d. \tag{8.6}$$

The volume spanned by the vectors $\hat{\boldsymbol{z}}$ is upper-bounded by the volume of vectors with Gaussian density of same covariance.

$$\mathrm{vol}(\hat{\boldsymbol{Z}}) \le \sqrt{(2\pi e)^d \det \hat{\boldsymbol{\Sigma}}} \tag{8.7}$$

Similarly, the volume spanned by the uniform noise is

$$\mathrm{vol}(\boldsymbol{N}) = \left(\frac{2s}{\sqrt{d}}\right)^d = \sqrt{\det \frac{(2s)^2}{d}\boldsymbol{I}_d}. \tag{8.8}$$

The number of bits to represent each vector $\boldsymbol{z}$ satisfying the ℓ_2-error bound ϵ^2 can be found as the number of bits to represent the index of a sphere spanned by uniform noise, packed in the region spanned by $\hat{\boldsymbol{\Sigma}}$. Therefore, we express an upper bound for average rate of a sample $\boldsymbol{z}$ in bits at a distortion ϵ under uniform quantization noise as

$$R(\boldsymbol{Z}|\epsilon) = \frac{1}{2}\log_2 \det\left[\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right] + \frac{d}{2}\log_2\left[\frac{2\pi e}{12}\right]. \tag{8.9}$$

Note that the rate depends only on singular values of $\boldsymbol{Z}$. Compared to the rate under Gaussian noise given in Eq. (8.2), the upper bound for the rate under quantization noise only has an additional linear term of d.

8.3.2 *Learning to Group*

Similar to Sec. 8.2.1, the set $\boldsymbol{Z}$ can be partitioned to k subsets $\boldsymbol{Z}_1 \cdots \boldsymbol{Z}_k$ according to a membership set $\boldsymbol{\Pi}$ of diagonal matrices $\{\boldsymbol{\Pi}_1, \cdots, \boldsymbol{\Pi}_k\} \in \mathbb{R}^{m \times m}$. Each diagonal element of $\boldsymbol{\Pi}_j(i,i)$ denotes the probability of sample i belonging to group j. Note that each sample can only belong to one group [69]:

$$\boldsymbol{\Pi}_j(i,i) = \pi_{ij} \in [0,1], \quad \textstyle\sum_{j=1}^{k} \pi_{ij} = 1.$$

Group j contains $m_j = \operatorname{tr}(\boldsymbol{\Pi}_j)$ samples. Similar to Eq. (8.3), for $m \gg d$, an upper bound for average rate for the given grouping $\boldsymbol{\Pi}$ can be written as

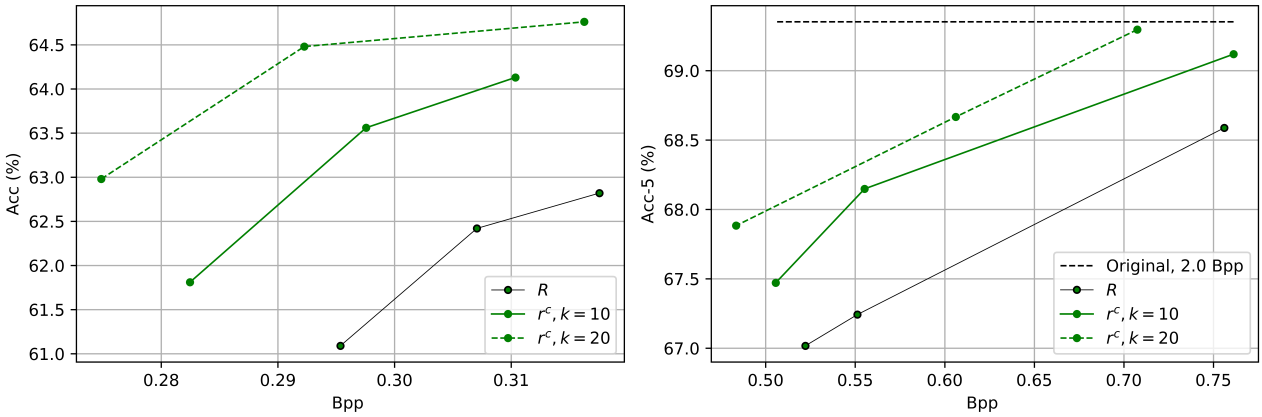
$$R^c(\boldsymbol{Z}|\epsilon, \boldsymbol{\Pi}) = \sum_{j=1}^{k} \frac{\operatorname{tr}(\boldsymbol{\Pi}_j)}{2m} \log_2 \det \left[\boldsymbol{I} + \frac{d}{\operatorname{tr}(\boldsymbol{\Pi}_j)\epsilon^2} \boldsymbol{Z}\boldsymbol{\Pi}_j \boldsymbol{Z}^\top \right] + \frac{d}{2} \log_2 \left[\frac{2\pi e}{12} \right]. \tag{8.10}$$

From (8.10), an optimal grouping $\boldsymbol{\Pi}^*$ can be learned to minimize the average rate-distortion for each sample $\boldsymbol{z}$. Thus, we formulate a rate minimization problem for given distortion ϵ:

$$\boldsymbol{\Pi}^* = \arg\min_{\boldsymbol{\Pi}} R^c(\boldsymbol{Z}|\epsilon, \boldsymbol{\Pi}) \tag{8.11}$$

We extend this rate minimization to an end-to-end optimization problem in conjunction with a learning task. Consider image classification using model $g(\cdot, \boldsymbol{\phi})$ that takes the reconstructed vector $\hat{\boldsymbol{z}}$ as the input and generates output label $y = g(\hat{\boldsymbol{z}}, \boldsymbol{\phi})$ as shown in Eq. (8.1). We propose to learn $\boldsymbol{\Pi}^*$ via end-to-end optimization of encoder f and learning model g. For this purpose, we minimize the following loss function.

$$\boldsymbol{\Pi}^*, \boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\Pi}, \boldsymbol{\theta}, \boldsymbol{\phi}, \epsilon} \mathbb{E}[\mathcal{L}_{\text{CL}}(y, y_{gt})] + \lambda \cdot R^c(\boldsymbol{Z}|\epsilon, \boldsymbol{\Pi}) \tag{8.12}$$



(a) CIFAR-100 (b) ImageNet-1k (size 32)

Figure 8.3: Rate-accuracy performance for CIFAR-100 ($d = 768$) and ImageNet-1k ($d = 1024$) data sets under quantization noise.

$\mathcal{L}_{\text{CL}}(y, y_{gt})$ denotes the classification cross entropy loss between the inferred class label y and the ground truth label y_{gt}. Here, the first RHS term of Eq. (8.12) is the average classification loss. λ is the Lagrangian to manage the trade-off between the rate and task accuracy [78, 104, 153]. $\boldsymbol{\theta}^*$ and $\boldsymbol{\phi}^*$ denote the learned encoder and classifier parameters, respectively.

We also propose to make the distortion variable ϵ trainable. This relaxation enables encoder more degree of freedom to adjust rate and shifts the role of controlling rate of latent vectors to parameter λ. On the other hand, since ϵ is related to the quantization parameter s, this allows the network to optimize the quantization interval as well.

8.4 EXPERIMENTS AND RESULTS

In this section, we describe the experimental setup and present test results from CIFAR-10 [111], CIFAR-100 [112] and ImageNet-1k [77] data sets. See supplement material in Appendix A for details on data sets. ImageNet-1k (size 32) consists of 1000 classes, each containing up to 1300 training images and 50 validation images resized to 32×32.

8.4.1 *DL Network Architectures*

Fig. 8.2 describes the encoder and classifier architectures in use. To generate the profile index j, we used a simple fully-connected layer with k nodes and assigned the index of the largest node as j.

For experiments that include a bandwidth/rate constraint, we initialized the training of end-to-end encoder-classifier framework shown in Fig. 8.1 by using pre-trained models, trained without a rate constraint, i.e., pre-trained models that were optimized for classification only. In CIFAR-10 and CIFAR-100 training, we fine-tuned the model using a "stochastic gradient decent" optimizer with a initial learning rate of 0.05. We reduced the learning rate each time by $\times 0.1$ at 10, 20, and 30 epochs until termination at 40 epochs. Similarly for ImageNet-1k (size 32), we used the same optimizer starting from a learning rate of 0.05, reduced each time by $\times 0.1$ at 5 and 10 epochs, respectively, until termination at 15 epochs[1].

8.4.2 *Quantization Noise Emulation*

We added random uniform noise to the vector $\boldsymbol{z}$ to generate $\hat{\boldsymbol{z}}$ as the input to the classifier according to Sec. 8.3.1. Following the work [104], we applied rounding function during the forward pass of the training and no quantization during the backward pass to approximate the loss function gradient. Consider an element z of the vector $\boldsymbol{z}$. We can write the differentiable quantization operation that maps z to quantized $\hat{z}$ in *Pytorch* as

$$\hat{z} = \text{torch.round}(\frac{z}{S}) \times S - z.\text{detach}() + z, \tag{8.13}$$

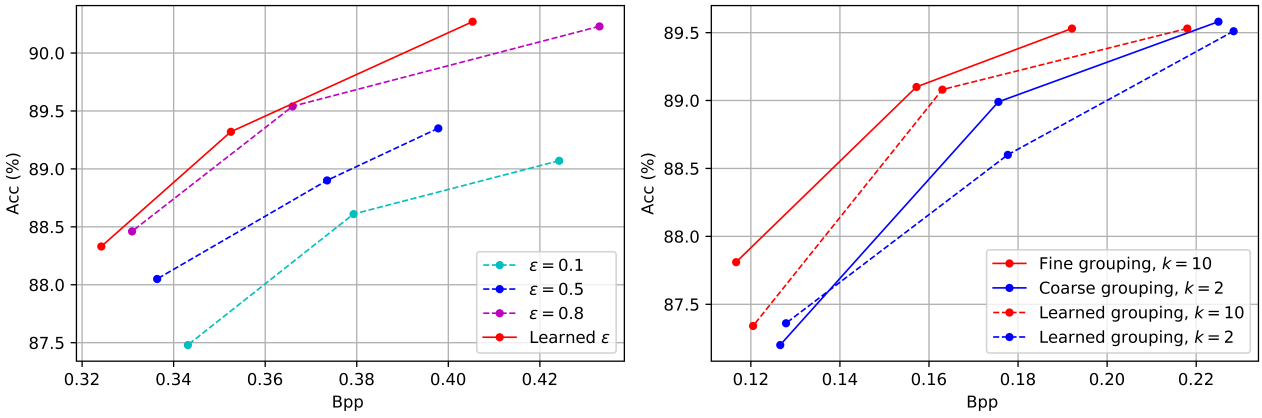
where $S = 2s/\sqrt{d}$ is the stepsize in this implementation.

[1] Source code is given at https://github.com/chamain/Learning-to-group.

Fig. 8.3 provides the results under quantization noise for CIFAR-100 and ImageNet-1k (size 32) data sets respectively for 10 and 20 coding profiles. We record the rate in bits per pixel (Bpp). With 20 learned coding profiles CIFAR-100 shows over 3.5% classification accuracy improvements over no grouping at 0.295 Bpp. Similarly, ImageNet-1k shows over 1% top-5 accuracy improvements at 0.525 bpp with 20 coding profiles over no grouping. We further note that increasing the number of coding profiles improves rate-accuracy tradeoff for both data sets particularly at lower data rates.

8.4.3 *Ablation Experiment: Distortion Learning*

In this study, we test trainable distortion parameter ϵ. By making ϵ trainable, we allow the network to optimize quantization stepsize s. To observe the benefit of trainable distortion ϵ, we also generate rate-accuracy points by optimizing the proposed framework for fixed ϵ values, as baselines. From the results in Fig. 8.4(a), we see that trainable ϵ makes it easier to change rate without having to set ϵ manually for each rate. Further, trainable ϵ produces comparably good results to fixed ϵ case.



(a) Learned distortion

(b) Learned grouping

Figure 8.4: (a) Ablation results using CIFAR-10 ($d = 1024$) data set without grouping (k=1) (b). Ablation results using CIFAR-10 ($d = 1024$) data set. Grouping can be learned to produce close $\mathcal{R}$-$\mathcal{C}$ performance to the fixed given groupings.

8.4.4 *Ablation Experiment: Learned Grouping*

To see how close the learned grouping is to ground truth, we set up two experiments to generate two baselines. For fixed grouping with fine labels ($k = 10$), during training and inference, we assumed that fine labels are available as grouping information. For instance all dog images of CIFAR-10 are assigned the same group index. For fixed grouping with coarse labels, during training and inference, we assumed that the coarse labels "Natural/Man-made"‘ for CIFAR-10 samples are available. This coarse labeling has been used in [159] and was shown to be helpful in classification. Here to compare the performance with the above baselines, we implement the proposed model with trainable $\mathbf{\Pi}$ for $k = 10$ and $k = 2$. See Fig. 8.4(b).

8.4.5 *Gaussian noise*

The performance gains achieved with the proposed grouping and end-to-end optimization are consistent under Gaussian noise. In this experiment, we added random Gaussian noise to $\boldsymbol{z}$ to generate noisy $\hat{\boldsymbol{z}}$ as the input to classifier. To train the network, we used the same function of Eq. (8.12) and updated the rate term $R^e(\boldsymbol{Z}|\epsilon, \mathbf{\Pi})$ for Gaussian noise as in Eq. (8.3). Considering practical applications, we fixed ϵ at 0.5 during the training. For the encoder-classifier model, we used the same optimizer, learning rates and scheduling as described in Sec. 8.4.1.

For ImageNet-1k (size 32) data set, results from Fig. 8.5 confirms the rate-accuracy performance gains with the latent dimensions $d = 1024$ for coding profiles of 10 and 20, respectively. With 20 learned coding profiles, the proposed method achieves over 2% top-5 classification accuracy improvement at 0.330 Bpp. We further note that these results of rate-accuracy performance improvement are consistent with results from quantization noise tests.

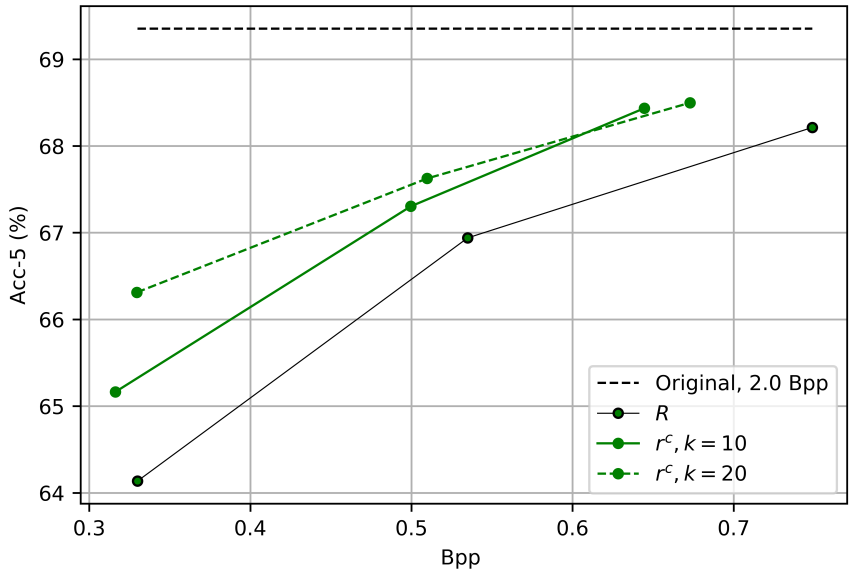


Figure 8.5: Rate-accuracy performance for ImageNet-1k ($d = 1024$) data set under Gaussian noise.

8.5 CONCLUSIONS

In this chapter, we propose a grouping-based end-to-end compression and classification framework for distributed learning involving low cost sensing devices. Based on CIFAR and ImageNet data sets, we observed considerable rate-classification accuracy improvements with learned grouping compared to no grouping case under quantization and Gaussian noise cases. The achieved rate-accuracy gains with learned grouping increase with number of grouping profiles at the cost of higher encoder complexity. We further note that the proposed architecture is computationally simple and easily trainable. We observe the importance of exploring more effective methods to accurately generate coding profile index. Equally important is the work to define discriminative group labels that are also simultaneously compressive.

9

CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize the contributions of this dissertation and discuss possible future research directions.

9.1 SUMMARY OF CONTRIBUTIONS

In this dissertation, focusing on image classification as the deep learning task of interest, we discussed our proposed approaches to improve rate-accuracy ($\mathcal{R}$-$\mathcal{C}$) and rate-distortion-accuracy ($\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$) performance of image compression codecs under bandwidth limited cloud inference scenarios. In Chapters 1-2, we laid the foundation for the importance of cloud-based image compression targeting deep learning tasks and its challenges. As the first part of the dissertation, in Chapters 3-4, we utilized JPEG2000 compressed images for faster and more accurate deep learning and discussed how to optimize JPEG2000 codec for rate-accuracy performance under limited bandwidth. As the second part of this dissertation, exploring end-to-end optimization of learning-based codecs, in Chapters 5, we proposed a novel classifier architecture based on variational auto-encoders (VAE) that outperforms several popular conventional image codecs in $\mathcal{R}$-$\mathcal{C}$ performance. Further investigating learning-based codecs, Chapters 6 discussed how to achieve better $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance based on ROI concept with end-to-end training. As the last part of this dissertation, Chapters 8-7 focused on leaning interpretable, efficient and low dimensional intermediate representation of images that improve $\mathcal{R}$-$\mathcal{C}$ performance of cloud-based networked AI applications. Exploring this direction, we developed Linear Discriminative Representation (LDR)-based image classifiers targeting zero-shot and few-shot learning problems.

9.1.1 *Open source codes*

The deep learning models presented throughout this dissertation are lighter-weight, faster and power efficient. We provide the GitHub repositories with python implementations of the proposed models in this dissertation below.

- Chapter 3 is available at chamain/Faster-JPEG2000-classification
- Chapter 4 is available at chamain/QuanNet
- Chapter 5 is available at chamain/VAE-classifier
- Chapter 6 is available at chamain/Classification-guided-ROI
- Chapter 7 is available at chamain/LDR-guided-Low-Shot-Learning
- Chapter 8 is available at chamain/Learning-to-group

9.2 LIMITATIONS AND FUTURE WORK

9.2.1 *Re-training/fine-tuning is necessary for each rate-accuracy point*

When achieving each rate-accuracy performance point of QuanNet in Chapter 4 and VAE-based classifier in Chapter 5, one has to save the deep learning model for each classification accuracy and compression ratio combination imposed by the trade-off parameter λ. Hence, in practical cloud inference scenarios, source device should be aware of the accuracy requirement of the application and send the bit stream of the image quantized with the matching step sizes learned from QuanNet in order to obtain the required accuracy. At the cloud end, the classifier model trained with corresponding step sizes should be used to achieve the proposed accuracy improvements. Similarly, in the case of VAE-based classifier, both the classifier model and the probability estimation need to be stored at the receiving end.

This requires storing several deep CNN models for each λ value, which can be memory consuming. This is valid for models trained on RGB images as well. For instance, to obtain the classification accuracies for different compression ratios in RGB domain, we trained a separate deep CNN for each compression ratio to obtain the RGB curve in Fig. 4.8. Using a single model training with high quality RGB images to perform inference for different compression ratios causes worse performance in test accuracy. See Fig. 9.1.

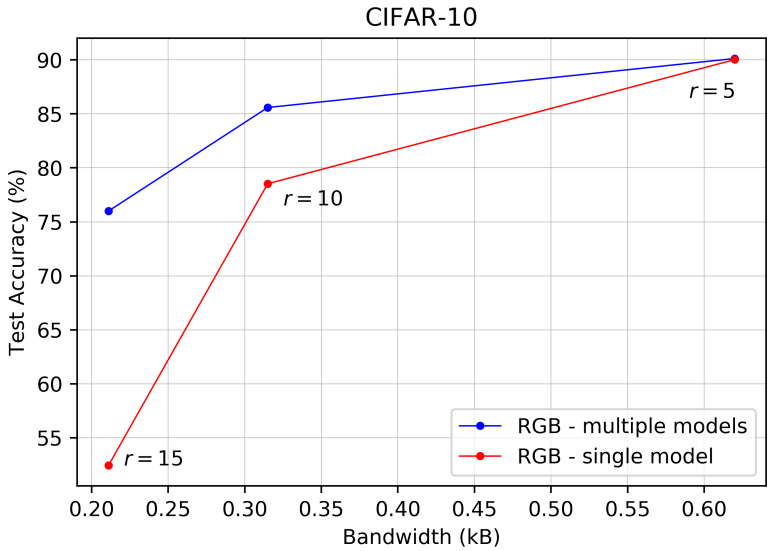


Figure 9.1: Accuracy comparison between single and multiple models. Red: uses model 'b' in Table 3.2 trained with high quality images compressed at $r = 0$ and tested for images compressed at $r = 5, 10, 15$. Blue: trained at tested at $r = 5, 10, 15$ separately.

Ideally, we prefer to have a single model trained that can give the proposed accuracies for different optimal parameter settings corresponding to different rates. In this direction, the authors of [106] propose a recurrent NN (RNN) that compresses images for classification with off-the shelf classifiers. In this case, classification network is not trained and their model is expected to learn important features for machine perception and compress accordingly. This may not give the best classification-compression characteristics possible, but uses the same classifier for inference for different image distortion levels. Therefore, exploring how to adapt model parameters, that are pre-trained for a given rate-accuracy, to a different rate-accuracy performance point during inference is an important future research direction.

9.2.2 *Universal encoders*

The bandwidth constraints and latency requirements of practical networked AI applications demand the data collected and transmitted by source devices to a cloud/edge node to be used for multiple DL inference applications such as visualization, classification, segmentation etc. In conventional cloud-based inference, since image data is compressed using rate-distortion optimized codecs, transmitted images can be reconstructed in RGB format at the cloud nodes before the inference. Since the majority of DL models are trained on RGB images, multiple DL applications can be performed on these RGB data using off-the-shelf DL models. However, as we showed in Chapters 3, 4, 5 and 6, such reconstruction of RGB images is unnecessary hence high latency, and off-the-shelf models are not rate-accuracy optimal demanding high bandwidth at a given accuracy.

When a model is trained end-to-end with a given rate and task loss for one DL application, the latent features learned from that model (at the encoder) poorly supports a second DL application [162]. Ideally, an encoder that transforms image data to a latent feature representations that support multiple DL applications can address this issue while enjoying the low latency, low bandwidth and high accuracy benefits of the proposed end-to-end models. We propose that features learned in self-supervised manner without directly depending on the end DL task can potentially become a universal encoder for multiple tasks. As future work, self-supervised feature learning based on linear discriminative representations [52, 53] will be a worthy pursuit.

A

DATA SETS

In this chapter, we provide details on the data sets we used during experiments related to this dissertation.

A.1 CLASSIFICATION DATA SETS

A.1.1 *ImageNet-1k*

The well-known ImageNet-1k [163] in classification consists of RGB images of size over 256×256 belonging to 1000 classes, each of which contains nearly 1300 training and 50 validation samples. We center-cropped images to size 256×256 during validation. We assess the performance of classification using top-5 accuracy.

A.1.2 *Imagenette*

It is a smaller subset of ImageNet-1K [77] data set consisting of the same image size belong to only 10 classes[1]: tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute. Each class contains about 1300 training and 50 validation samples. We also center-cropped images to size 256×256.

[1] Imagenette can be found at: https://github.com/fastai/imagenette

A.1.3 *Tiny-ImageNet*

Tiny ImageNet is a subset of ImageNet data set [77]. It consists of RGB images of size 64 × 64 belonging to 200 classes, each class with 1300 training images and 50 validation images.

A.1.4 *ModelNet-40*

ModelNet-40 [164] is a well known data set for 3D object classification that contains 12,311 3D CAD models belonging to 40 classes. We can use these models to sample point clouds and multi views for each object. Further, ModelMet-40 provides 2D images of 12 view points per object in each class.

A.1.5 *CIFAR-10*

CIFAR-10 [111] consists of 10 classes:. Each class has 5000 images for training and 1000 images for testing per class. CIFAR-10 images are in RGB format of size 32×32.

A.1.6 *CIFAR-100*

Similar to CIFAR-10, CIFAR-100 [112] data set contains 50k training and 10k test RGB images of size 32×32 in 100 classes.

A.1.7 *AwA2*

AwA2 [147] contains 37,322 images of 50 animal classes with 85 attributes per class. We used the standard training and test (40 seen/10 unseen) split proposed in [54].

A.1.8 *CUB*

CUB [148] contains 11,788 bird images from 150 seen and 50 unseen classes with 312 attributes per class.

A.1.9 *SUN*

SUN [149] contains 14,340 scene images from 645 seen and 72 unseen classes, with 102 attributes per class.

A.2 COMPRESSION DATA SETS

A.2.1 *HKU-IS*

This visual saliency prediction data set [165] consists of larger training and test images (above 256×256 in size). Most of these images contain multiple saliency objects. We used HKU-IS data to evaluate the ROI prediction accuracy of the proposed ROI coding module.

A.2.2 *DAVIS-2016*

Densely Annotated Video Instance Segmentation (DAVIS)-2016 [166] is a popular video object segmentation data set of 30 training and 20 test videos available in RGB image frames of size 640×480 along with annotated masks for single object segmentation. We categorized frames of test set into 3 groups according to the ratio of ROI mask area to the full image area: 5%, 24% and 36%.

A.2.3 *CADDY*

Cognitive autonomous diving buddy (CADDY) data set [167] is an underwater gesture data set of RGB images collected in open sea, indoor, and outdoor pools. Its image samples are of size 640×480 belonging to 16 gesture classes. Following the approach in [168], we pseudo-randomly split the data into training set (70%) and test set (30%) based on a selected seed. For faster batch training, we resized the images to 341×256 without changing the aspect ratio.

A.2.4 *ASL*

American Sign Language (ASL) data we used in this work is a manually collected RGB image data set available online[2]. It consists of 8442 raw RGB images of size over 640×480 belonging to 24 character classes: English alphabet except "J" and "Z" (characters with motion). Similar to CADDY data set, we pseudo-randomly spilt ASL data set into 70% training and 30% test sets based on a fixed random seed (30). For faster batch training, we resized the images to have a minimum dimension of 256 without changing the aspect ratio.

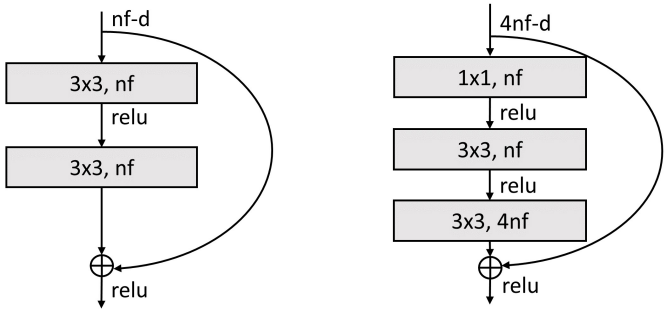
[2] ASL data set can be found at
https://www.kaggle.com/signnteam/asl-sign-language-pictures-minus-j-z

B

FASTER AND ACCURATE DEEP LEARNING FOR JPEG2000 IMAGES

B.1 RESNET FOR CLASSIFICATION

In this dissertation, we mainly use ResNet and some variations of ResNet as the deep CNN for image classification tasks. There are two building blocks of ResNet: a 'Basic' building block for shallow models like ResNet 18, 34 and a 'Bottleneck' block for deeper models like ResNet 50, 101 and 152 [15]. Figure B.1 illustrates these building blocks.



(a) Basic (b) Bottleneck

Figure B.1: Building blocks for ResNet. nf is the number of filters. Both blocks have the same time complexity. (a) is used in ResNet 18, 34. (b) is used in ResNet 50, 101 and 152.

For the experiments in Part i, we used CIFAR-10 [111] and ImageNet [77] data sets for 2D image classification. Table B.1 and Table B.2 summarizes the baseline architectures for CIFAR-10 and ImageNet.

Table B.1: ResNet architecture for CIFAR-10. Basic blocks with different nf are used as the building blocks.

layer name	output size	8-layer	20-layer
conv1	32×32	3×3, 16 stride 1	
conv2x	32×32	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 3$
conv3x	16×16	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$
conv4x	8×8	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$
	1×1	avg pool 10-d fc softmax	

Table B.2: ResNet architecture for ImageNet. Basic and bottleneck blocks with different nf are used as the building blocks for 18-layer and 50-layer ResNets.

layer name	output size	layer-8	layer-20
conv1	112×112	7×7, 64 stride 2	
conv2x	56×56	3×3, max pool stride 2 $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
conv5x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	avg pool 1000-d fc softmax	

B.2 FINE-TUNING ON PRE-TRAINED MODELS

Fine tuning a base model for limited bandwidth can reduce training time. Figure B.2 shows that the pre-training is feasible and achieves a modest accuracy gain for both RGB and CDF 9/7 DWT coefficients of the CIFAR-10 images.

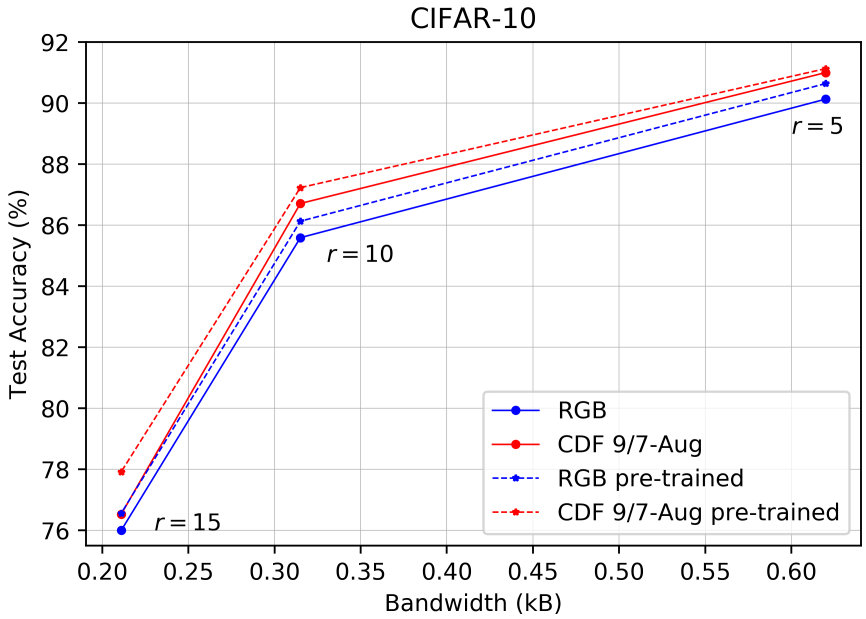


Figure B.2: Fine tuning for $r = 5, 10, 15$ using the pretrained DWT model for $r = 0$ reduces training time by 75% and improves accuracy. This observation is consistent with pre-training in RGB domain. CDF and RGB inputs use model 'b' and 'e' respectively.

B.3 3D MODEL CLASSIFICATION WITH MULTI-VIEW CNNS

In order to further validate our claims, we experimented with 3D model classification with JPEG2000 compressed images. 3D object classification has gained popularity in parallel to 2D image classification with deep neural networks over the past few years. We can identify several popular approaches for 3D object classification based on the input format. For instance, a 3D object can be represented as a point cloud, a volumetric object or using multi-view 2D

images. ModelNet-40 [164] is a well known data set for 3D object classification that contains 12,311 3D CAD models belonging to 40 classes. We can use these models to sample point clouds and multi views for each object.

A recent work PointNet [169] suggests a classification network with symmetric feature operations like max pooling to classify un-ordered point clouds. The authors of MVCNN [170] use 2D images of 12 view points per object and train a CNN to extract view based features for 3D object classification. They use CNNs to obtain features for each view point, pool the view based features and use a softmax classifier for classification. See Figure B.3.

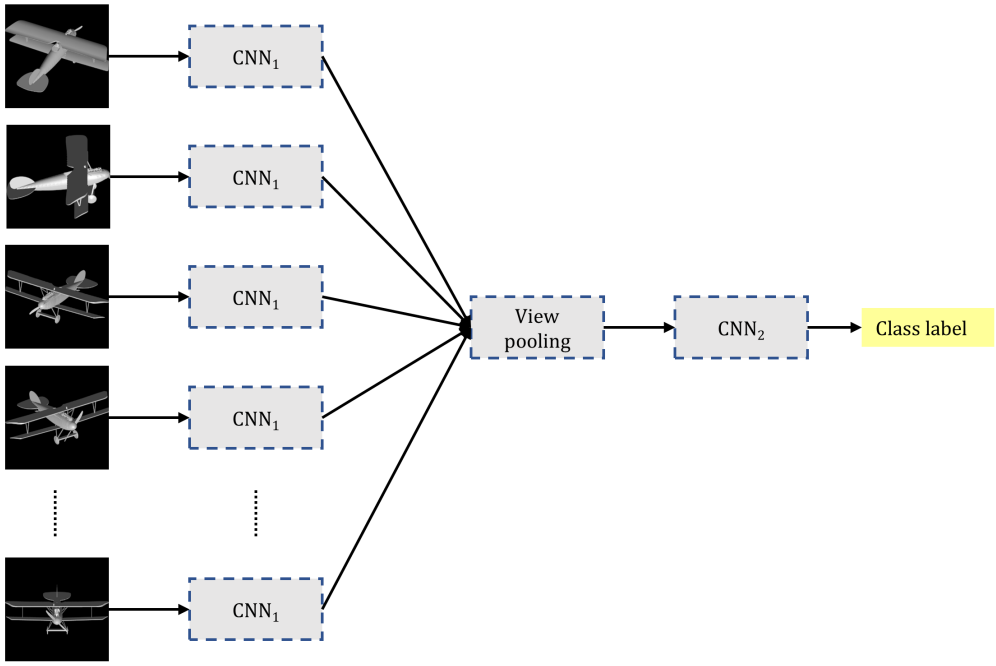


Figure B.3: Multi View CNN (MVCNN) model. At test time, a 3D shape is rendered from 12 different views and are passed thorough CNN_1 to extract view based features. These are then pooled across views and passed through CNN_2 to obtain a compact shape descriptor. [170]

We used ResNet-18 [15] as the CNNs in Figure B.3 and used that model as the baseline model for RGB inputs encoded and decoded as JPEG2000. Then we extracted DWT level-1 coefficients as discussed in section 3.2 and used them as the input for MVCNN. We observe the similar accuracy and training/inference improvements with DWT inputs even before considering reconstruction savings. See table B.3. We used same training and test split as in

[164] and [170]. Input images are 224×224 RGB images for the baseline and 112×112×12 for DWT model.

Table B.3: Results of ModelNet-40.

parameter	RGB	DWT (ours)
Overall test Acc. (%)	93.15	**93.96**
Avg. class test Acc. (%)	90.04	**91.41**
No of CONV layers	20	9
No of parameters (M)	11.20	8.87
Training rate/epoch (images/s)	979	**1112**
Inference rate (images/s)	3042	**3050**

Classification in DWT domain shows over 1% accuracy improvement with faster training using over 20% less number of parameters compared to RGB domain on ModelNet-40 data set. We summarize the ResNet models we used in the table B.4. During training we used Adam optimizer with 5e-5 learning rate for 60 epochs with 0.001 weight decay and batch size of 8×12 (3D models×no of views) without pre-training.

Table B.4: ResNet architecture for ModelNet-40.

layer name	RGB		DWT	
	output size	operation	output size	operation
conv1	112×112	7×7, 64 stride 2	56×56	7×7, 256 stride 2
		3×3, max pool stride 2		
conv2x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	28×28	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 1$
conv3x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	14×14	$\begin{bmatrix} 3\times3, 384 \\ 3\times3, 384 \end{bmatrix} \times 1$
conv4x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	7×7	$\begin{bmatrix} 3\times3, 576 \\ 3\times3, 576 \end{bmatrix} \times 1$
conv5x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	-	-
	1 × 1	avg pool 40-d fc softmax		

Compared to 2D image classification with CIFAR-10 and imageNet, we were able to reduce the number of parameters significantly for DWT models. Inputs being in the gray

scale can be the result for this observations. In comparison to RGB inputs, gray scale inputs contain less number of features hence less number of parameters may be sufficient to capture the input variations of the compact DWT format.

B.4 RESNET ARCHITECTURES USED FOR DB1, DWT COEFFICIENTS.

Table B.5: ResNet architectures used for DB1, DWT coefficients.

layer name	output size	8-layer	20-layer
conv1	16×16	3×3, 16 stride 1	3×3, 50 stride 1
conv2x	16×16	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 50 \\ 3 \times 3, 50 \end{bmatrix} \times 3$
conv3x	8×8	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 100 \\ 3 \times 3, 100 \end{bmatrix} \times 3$
conv4x	4×4	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 200 \\ 3 \times 3, 200 \end{bmatrix} \times 3$
	1×1	avg pool 10-d fc softmax	

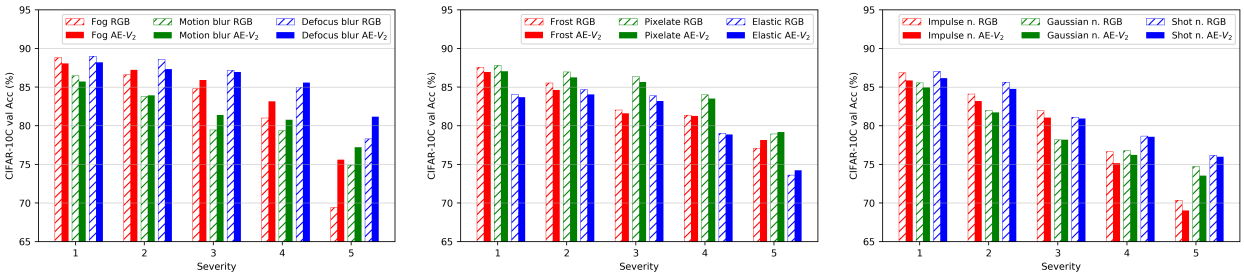
C

VAE-BASED CLASSIFIERS

C.1 ROBUSTNESS TO VISUAL CORRUPTIONS

In this section, we demonstrate the robustness of the proposed VAE-based classifier against common visual corruptions of images based on CIFAR-10-C [171] data set. CIFAR-10-C data set provides corrupted versions of CIFAR-10 [111] test data set for visual corruptions such as noise, blur, pixelation etc. at 5 different severity levels of each corruption.

We selected the proposed AE-V_2 VAE model with $L = 6$ trained on CIFAR-10 data as discussed in Sec. 5.4.1 during inference on CIFAR-10-C test data set. For 9 types of corruptions namely **Fog, Motion Blur, Defocus Blur, Frost, Pixelation, Elastic transformation, Impulse noise, Gaussian noise and Shot noise**, we repeated the above inference for classification at 5 severity levels. Figures C.1(a)-(c) show the results. 'RGB' baseline corresponds to the inference accuracy with ResNet-18 classifier trained on CIFAR-10 training set.



(a) fog, motion blur and defocus (b) frost, pixelation and elastic (c) impulse, Gaussian and shot noise

Figure C.1: Robustness comparison for visual corruptions: fog, motion blur, defocus blur, frost, pixelation, elastic transformation, impulse noise, Gaussian noise, shot noise on CIFAR-10-C data set.

We note that AE-V_2 classifier outperforms ResNet-18 baseline classifier at inference on images corrupted with fog, motion blur and defocus blur at 3-5 severity levels by a significant

margin. Further the proposed model performs slightly worse than the baseline classifier on images corrupted with impulse, Gaussian and shot noise.

C.2 IMPLEMENTATION DETAILS: EFFECT OF β ADJUSTMENT

Recall that the VAE loss of the proposed method for a sample $\boldsymbol{x}$ is given by,

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}, y_{gt}) = \mathcal{L}_{\mathrm{CL}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}, y_{gt}) + \beta \mathcal{L}_{\mathrm{R}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}) \tag{C.1}$$

where $\mathcal{L}_{\mathrm{R}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x})$ is the bandwidth of $\mathbf{z}$ transmitted over a network link. We measure the bandwidth (rate) in BPP in our experiments. The effect of the trade-off parameters β for CIFAR-10 data set is shown in Fig. C.2(a).

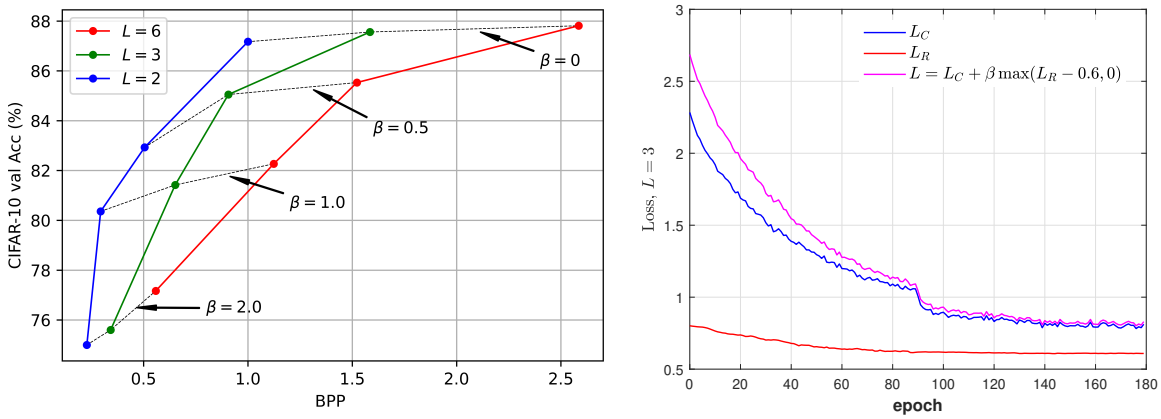
In order to achieve optimal rate-classification accuracy performance at a given rate (r_t), we minimize the following loss instead of the loss given in Eq. (C.1).

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}, y_{gt}) = \mathcal{L}_{\mathrm{CL}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}, y_{gt}) + \beta \max\left(\mathcal{L}_{\mathrm{R}(\boldsymbol{\theta},\boldsymbol{\phi})}(\hat{\mathbf{z}}|\boldsymbol{x}) - r_t, 0\right) \tag{C.2}$$

We observed better classification accuracy by this approach at a rate slightly above the requirement r_t. In Fig. C.2(b) we illustrate the loss curves for a selected β and an r_t value.

C.3 VISUALIZATION OF LATENT MAPS

Figs. C.3(a), (b) and (c) show the latent maps $\hat{\mathbf{z}}$ of a sample image $\mathbf{x}$ for $L = 6, 3, 2$ related to the model AE-V_3. For each sub figure, top-left color image is the original and bottom left color image is the reconstructed from the latent maps visualized next to them. We visualize 32 latent maps (each of size 20×20) before ($\mathbf{z}$) and after quantization maps ($\hat{\mathbf{z}}$). Starting

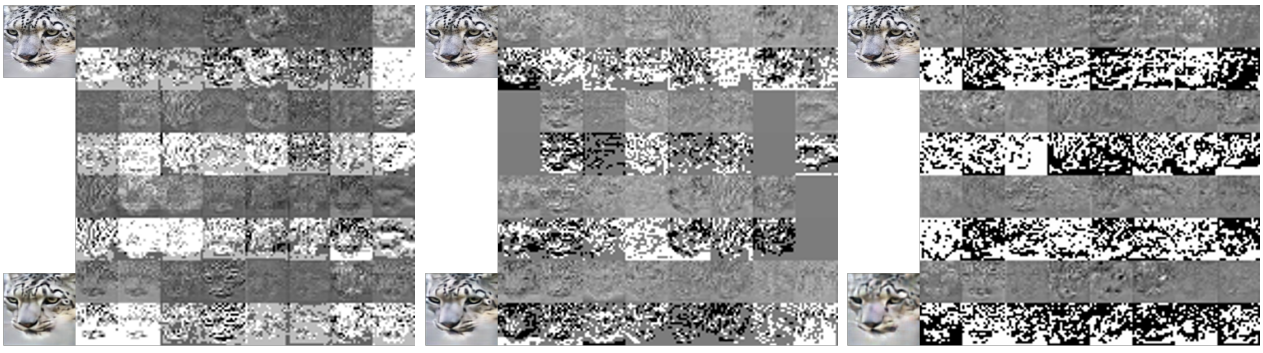


(a) Effect of β (b) Convergence of losses

Figure C.2: (a). CIFAR-10 results for different β values. Black dash lines connect the points for AE models with the same β value. (b). CIFAR-10 average validation losses obtained with Eq. (C.2) at $\beta = 2$ and $r_t = 0.6$ for AE-V$_1$, $L = 3$ model. Note the smooth convergence of the rate L_C around 0.6 at higher number of epochs.

from the first row, the $\mathbf{z}$ maps are given in every other row. The corresponding quantized $\hat{\mathbf{z}}$ is given right below each $\mathbf{z}$ map. Each map is normalized before visualization.

Observe that for each number of quantization centers L, the quantized maps $\hat{\mathbf{z}}$ has only L different colors. Some of the $\hat{\mathbf{z}}$ maps for $L = 6$ show clearly identifiable snow leopard figures implying that high level information of $\mathbf{x}$ is preserved during training and helpful for classification.



(a) $L = 6$ (b) $L = 3$ (c) $L = 2$

Figure C.3: Latent maps $\hat{\mathbf{z}}$ of a sample image $\mathbf{x}$ for $L = 6$, 3 and 2 related to the model AE-V$_3$. For each sub figure, top-left color image is the original and bottom left color image is the reconstructed from the latent maps visualized next to them. Some of the $\hat{\mathbf{z}}$ maps for $L = 6$ show clearly identifiable snow leopard figures implying that high level information of $\mathbf{x}$ is preserved during training and helpful for classification.

Furthermore, we demonstrate that such feature maps that are learned and transmitted by the proposed VAE for classification can also be used for reconstructing source images with sufficient quality in terms of achieved PSNR. We show that such retraining of reconstruction decoder model can be successful without compromising the performance of VAE-based classification.

C.4 FURTHER THEORETICAL EXPLANATIONS

C.4.1 *Relationship to the Information Bottleneck*

In this section, we theoretically derive the relationship between the proposed VAE framework for classification and the well-known Information Bottleneck principle [102, 103]. Consider the random variables $\mathbf{Y}$, $\mathbf{x}$ and $\mathbf{z}$ related according to a Markov chain as follows.

$$\mathbf{Y} \rightarrow \mathbf{x} \xrightarrow[\boldsymbol{\phi}]{\mathrm{E}} \mathbf{z} \tag{C.3}$$

In the context of image classification with auto-encoders $\mathbf{Y}$, $\mathbf{x}$ and $\mathbf{z}$ can be viewed as variables corresponding to image label, image and latent encoding of the image respectively. $\mathbf{z}$ is mapped from $\mathbf{x}$ with an Encoder ($\mathbf{E}$) parameterized by $\boldsymbol{\phi}$. Hence, we can denote $\mathbf{z}$ as $\mathbf{z}(\boldsymbol{\phi})$, which is often written as $\mathbf{z}$ for simplicity.

We can write the objective function of the Information Bottleneck (IB) principle [102, 103] as the following.

$$\max_{\boldsymbol{\phi}} \mathrm{IB}_{\mathbf{x},\mathbf{z},\mathbf{Y}}(\boldsymbol{\phi}) = I(\mathbf{z}(\mathbf{E}); \mathbf{Y}) - \beta I(\mathbf{x}; \mathbf{z}(\mathbf{E})), \quad \beta \geq 0 \tag{C.4}$$

$I(\mathbf{x};\mathbf{z})$ is the Mutual Information (MI) between the random variables $\mathbf{x}$ and $\mathbf{z}$. We rewrite this as a minimization objective to match the above VAE narrative.

$$\min_{\boldsymbol{\phi}} -I(\mathbf{z}(\text{Œ});\mathbf{Y}) + \beta I(\mathbf{x};\mathbf{z}(\text{Œ})), \quad \beta \geq 0 \tag{C.5}$$

Following the definition of MI we write,

$$I(\mathbf{z};\mathbf{Y}) = \int p(y,\boldsymbol{z}) \log \frac{p(y|\boldsymbol{z})}{p(y)} dy \; d\boldsymbol{z} \tag{C.6}$$

In order to track the distribution $p(y|\mathbf{z})$ we employ a classifier (**CL**) parameterized by $\boldsymbol{\theta}$ and estimate the conditional label distribution $\eta_{\boldsymbol{\theta}}(y|\mathbf{z})$ as in Eq. (5.3). Since $\text{KL}(p(\mathbf{Y}|\mathbf{z})|\eta_{\boldsymbol{\theta}}(\mathbf{Y}|\mathbf{z})) \geq 0$ we can write the inequality,

$$\int p(y|\boldsymbol{z}) \log p(y|\boldsymbol{z}) dy \geq \int p(y|\boldsymbol{z}) \log \eta_{\boldsymbol{\theta}}(y|\boldsymbol{z}) dy \tag{C.7}$$

We utilize this inequality to bound for $I(\mathbf{z};\mathbf{Y})$ in Eq. (C.6).

$$I(\mathbf{z};\mathbf{Y}) = \int p(y,\boldsymbol{z}) \log p(y|\boldsymbol{z}) dy \; d\boldsymbol{z} - \int p(y) \log p(y) dy \geq \int p(y,\boldsymbol{z}) \log \eta_{\boldsymbol{\theta}}(y|\boldsymbol{z}) dy \; d\boldsymbol{z} \; + \text{H}(\mathbf{Y}) \tag{C.8}$$

Leveraging the Markov assumption in Eq. (C.3) , $p(y,\boldsymbol{z},\mathbf{x}) = p(\mathbf{x},y) q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ with the encoder distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, we can express the bound in Eq. (C.8) for $I(\mathbf{z};\mathbf{Y})$ as the following.

$$\begin{aligned} I(\mathbf{z};\mathbf{Y}) &\geq \int p(y,\boldsymbol{z},\boldsymbol{x}) \log \eta_{\boldsymbol{\theta}}(y|\boldsymbol{z}) d\boldsymbol{x} \; dy \; d\boldsymbol{z} + \text{H}(\mathbf{Y}) \\ &\geq \int p(\boldsymbol{x},y) q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \eta_{\boldsymbol{\theta}}(y|\boldsymbol{z}) d\boldsymbol{x} \; dy \; d\boldsymbol{z} + \text{H}(\mathbf{Y}) \end{aligned} \tag{C.9}$$

For the second term of the IB objective given in Eq. (C.5), following the definition of MI, we write,

$$I(\mathbf{x};\mathbf{z}) = \int q_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{z}) \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z}\ d\boldsymbol{x} - \int q_{\boldsymbol{\phi}}(\boldsymbol{z}) \log q_{\boldsymbol{\phi}}(\boldsymbol{z}) d\boldsymbol{z} \tag{C.10}$$

With the definition of KL divergence, for an arbitrary $r(\mathbf{z})$, we write the following inequality similar to Eq. (C.7).

$$\int q_{\boldsymbol{\phi}}(\boldsymbol{z}) \log q_{\boldsymbol{\phi}}(\boldsymbol{z}) d\boldsymbol{z} \geq \int q_{\boldsymbol{\phi}}(\boldsymbol{z}) \log r(\boldsymbol{z}) d\boldsymbol{z}. \tag{C.11}$$

We combine Eq. (C.10) and (C.7) to obtain an upper bound for $I(\mathbf{x};\mathbf{z})$.

$$\begin{aligned} I(\mathbf{x};\mathbf{z}) &\leq \int q_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{z}) \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z}\ d\boldsymbol{x} - \int q_{\boldsymbol{\phi}}(\boldsymbol{z}) \log r(\boldsymbol{z}) d\boldsymbol{z} \\ &= \int q_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{z}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{r(\boldsymbol{z})} d\boldsymbol{z}\ d\boldsymbol{x} \\ &= \int p(\boldsymbol{x}) q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{r(\boldsymbol{z})} d\boldsymbol{z}\ d\boldsymbol{x} \qquad \text{(C.12)} \\ &= \int p(\boldsymbol{x},y) q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{r(\boldsymbol{z})} d\boldsymbol{x}\ dy\ d\boldsymbol{z} \qquad \text{(C.13)} \end{aligned}$$

Eq. (C.13) follows from Eq. (C.12) since $p(\boldsymbol{x}) = \int p(\boldsymbol{x},y) dy$. Similar to [172], we approximate $p(\boldsymbol{x},y)$ with an empirical distribution based on data samples $(\boldsymbol{x}_n, y_n) \in \mathcal{S}$ s.t.,

$$p(\boldsymbol{x},y) = \frac{1}{|\mathcal{S}|} \sum_{n=1}^{|\mathcal{S}|} \delta_{\boldsymbol{x}_n}(\boldsymbol{x}) \delta_{y_n}(y). \tag{C.14}$$

Hence the approximated IB loss for minimization can be written as the following.

$$\begin{aligned}
\mathrm{IB}_{\mathbf{x},\mathbf{z},\mathbf{Y}}(\boldsymbol{\phi},\boldsymbol{\theta}) \leq & -\int p(\boldsymbol{x},y)q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\log\eta_{\boldsymbol{\theta}}(y|\boldsymbol{z})d\boldsymbol{x}\ dy\ d\boldsymbol{z} - \mathrm{H}(\mathbf{Y}) \\
& + \beta\int p(\boldsymbol{x},y)q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\log\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{r(\boldsymbol{z})}d\boldsymbol{x}\ dy\ d\boldsymbol{z} \qquad \text{(C.15)} \\
\approx & \frac{1}{|\mathcal{S}|}\sum_{n=1}^{|\mathcal{S}|}\{-\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)\log\eta_{\boldsymbol{\theta}}(y_n|\boldsymbol{z})d\boldsymbol{z} \\
& + \beta\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)\log\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)}{r(\boldsymbol{z})}d\boldsymbol{z}\} - \mathrm{H}(\mathbf{Y}) \qquad \text{(C.16)} \\
= & \frac{1}{|\mathcal{S}|}\sum_{n=1}^{|\mathcal{S}|}\{-\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)\log\eta_{\boldsymbol{\theta}}(y_n|\boldsymbol{z})d\boldsymbol{z} \\
& + \beta\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)\log\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}_n)}d\boldsymbol{z}\} - \mathrm{H}(\mathbf{Y}) \qquad \text{(C.17)}
\end{aligned}$$

From Eq. (C.16) to Eq. (C.17), we replace the arbitrary $r(\boldsymbol{z})$ with $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}_n)$.

Using the definition of KL divergence, we re-write Eq. (C.17) as the following which is upper bounded by the β-VAE loss $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}$ for classification.

$$\begin{aligned}
\mathrm{IB}_{\mathbf{x},\mathbf{z},\mathbf{Y}}(\boldsymbol{\phi},\boldsymbol{\theta}) &\leq \frac{1}{|\mathcal{S}|}\sum_{n=1}^{|\mathcal{S}|}\{-\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)\log\eta_{\boldsymbol{\theta}}(y_n|\boldsymbol{z})d\boldsymbol{z} + \beta\ \mathrm{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x}_n), p_{\boldsymbol{\theta}}(\mathbf{z}|\boldsymbol{x}_n)]\} - \mathrm{H}(\mathbf{Y}) \\
&\leq \frac{1}{|\mathcal{S}|}\sum_{n=1}^{|\mathcal{S}|}\{\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x_n},y_n)\} - \mathrm{H}(\mathbf{Y}) \qquad \text{(C.18)}
\end{aligned}$$

A Similar result has been shown in [172] for image reconstruction.

C.4.2 *An alternative problem formulation*

Assuming the same setting as we discussed above, we provide an alternative derivation the loss function for joint image classification and compression as follows.

We start with the classification cross entropy loss between the true and estimated label distributions. From Eqs. (5.6)(5.7), we can write the cross entropy loss as

$$\mathrm{CE}_{y|\boldsymbol{x}}(\rho, \hat{\rho}_{\boldsymbol{\theta}}) = -\sum_{c \in \mathcal{Y}} \rho(y = c|\boldsymbol{x}) \log \hat{\rho}_{\boldsymbol{\theta}}(y = c|\boldsymbol{x}) - \log \hat{\rho}_{\boldsymbol{\theta}}(y_{\mathrm{gt}}|\boldsymbol{x}) \tag{C.19}$$

For the setting with an Encoder ($\mathbf{E}$) with the density $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, we have

$$\mathbf{x} \xrightarrow[\boldsymbol{\phi}]{\mathrm{E}} \mathbf{z} \xrightarrow[\boldsymbol{\theta}]{\mathrm{CL}} y. \tag{C.20}$$

Considering different instances of $\mathbf{z}$ for a given sample $\boldsymbol{x}$, Eq. (5.3) can be re-written to formulate the following problem to maximize $\hat{\rho}_{\boldsymbol{\theta}}(y_{\mathrm{gt}}|\boldsymbol{x})$ in order to minimize classification cross entropy.

$$\max_{\boldsymbol{\theta},\boldsymbol{\phi}} E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}} \left[\eta_{\boldsymbol{\theta}}(y_{\mathrm{gt}}|\mathbf{z})\right] \tag{C.21}$$

Since the density $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is not available at the Decoder ($\mathbf{D}$), we employ a Probability Estimator $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ to closely estimate $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. This introduces a constraint to the problem in Eq. (C.21) with $\epsilon \geq 0$ [68, 100].

$$\max_{\boldsymbol{\theta},\boldsymbol{\phi}} E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}} \left[\eta_{\boldsymbol{\theta}}(y_{\mathrm{gt}}|\mathbf{z})\right] \qquad \text{s.t.} \quad 0 \leq \mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}) \leq \epsilon \tag{C.22}$$

This can be re-written as a minimization problem by introducing a monotonic $-\log()$ function.

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} -\log E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}} \left[\eta_{\boldsymbol{\theta}}(y_{\mathrm{gt}}|\mathbf{z})\right] \qquad \text{s.t.} \quad \mathrm{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}) \leq \epsilon \tag{C.23}$$

According to Jensen's inequality, we can write,

$$-\log E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}} \left[\eta_{\boldsymbol{\theta}}(y_{\mathrm{gt}}|\mathbf{z})\right] \leq E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}} \left[-\log(\eta_{\boldsymbol{\theta}}(y_{gt}|\boldsymbol{z}))\right]. \tag{C.24}$$

Instead of directly minimizing the optimization problem in Eq. (C.23), we can minimize its upper bound as follows.

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}}\left[-\log(\eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\mathbf{z}))\right] \qquad \text{s.t.} \quad \text{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}) \leq \epsilon \tag{C.25}$$

By re-writing the problem in Eq (C.25) as a Lagrangian under KKT [173] conditions for the Lagrangian multiplier $\beta \geq 0$, we write,

$$\mathcal{F}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}, y_{gt}) = E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\left[-\log(\eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\mathbf{z}))\right] + \beta\left[\text{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z}|\mathbf{X},\boldsymbol{\theta}}) - \epsilon\right]. \tag{C.26}$$

With this we can arrive at the β-VAE loss function for classification.

$$\mathcal{F}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}, y_{\text{gt}}) \leq \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}, y_{\text{gt}}) = E_{q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}}\left[-\log(\eta_{\boldsymbol{\theta}}(y_{\text{gt}}|\mathbf{z}))\right] + \beta\text{KL}(q_{\mathbf{z}|\mathbf{x},\boldsymbol{\phi}}|p_{\mathbf{z}|\mathbf{x},\boldsymbol{\theta}}). \tag{C.27}$$

D

CLASSIFICATION-GUIDED ROI-BASED END-TO-END IMAGE COMPRESSION

D.1 POOLNET GENERATED ROI

Fig. D.1 illustrates the ROI predictions generated by PoolNet [117] for some selected samples of HKU-IS data set.

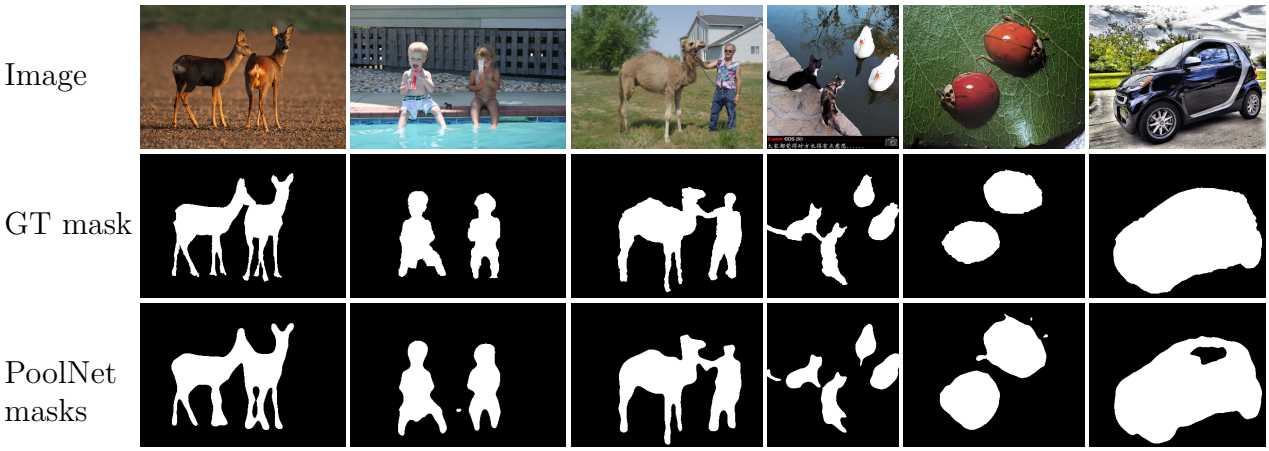


Figure D.1: ROI mask comparison for some selected samples of HKU-IS dataset. PoolNet [117] with ResNet-50 backbone generates ROI masks with mean absolute error (MAE) of 0.0362. GT: ground truth.

D.2 COMPLEXITY COMPARISON

In this section, we provide an detailed complexity comparison of the ROI prediction of proposed classification-guided ROI codec and PoolNet [117] (baseline) in terms of number of parameters. Our proposed feature-sharing ROI prediction achieves 81% (5 times) of parameter reduction compared to the baseline.

It is note worthy that when guided with classification loss, a simple encoder backbone is sufficient to extract relevant features for saliency map generation. Compared to ResNet-50

Table D.1: Complexity comparison: No. of parameters are given in millions (M). Our proposed feature-sharing ROI prediction achieves 81% (5 times) of parameter reduction compared to the baseline.

Module	PoolNet	Classification guided ROI	Percentage reduction
ROI prediction	68.26	**12.9**	81%
−Feature backbone	ResNet-50 25.51	Encoder (g_a) **1.49**	 94%
−Saliency generation	42.75	**11.41**	73%
– Pyramid pooling	11.27	**1.47**	87%
– Deep pooling	24.4	**8.83**	64%
– Aggregation	7.08	**1.11**	84%

backboned used in PoolNet, Our proposed ROI prediction reuses the features extracted from over 10 times less complex encoder backbone. More importantly, our proposed ROI codec achieves superior ROI rate-distortion performance compared to the baseline even after above such complexity reductions.

D.3 CAI-2019 COMPLEXITY ESTIMATION

When calculating number of parameters for Cai-2019 [38], we relied on the implementation details provided in their paper since implementation code is unavailable.

During the parameter estimation of Cai-2019, we did not include the parameters of the entropy coder and entropy decoder in Table. D.2. Note that the backbone image coder of Cai-2019 (Encoder+Decoder) alone has 10.91 M parameters which is 2 times the SHP codec we used as the backbone image codec.

D.4 CLASSIFIER FINE-TUNING

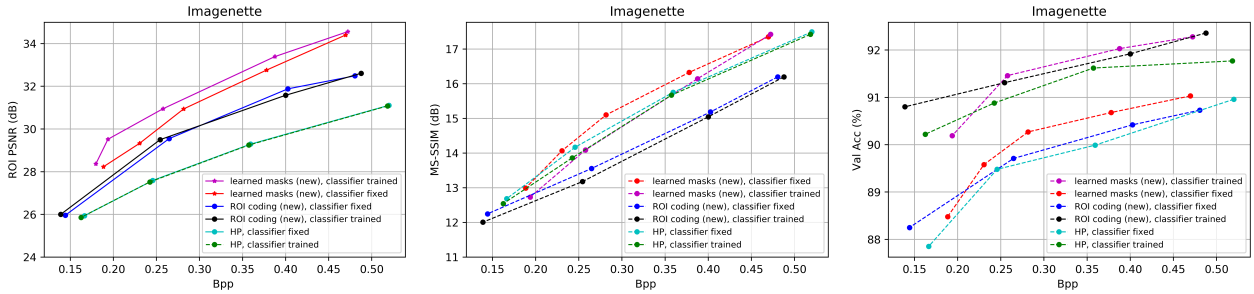
Thus far, when training the baselines and the proposed framework, we kept the image classifier frozen at the default pre-trained state. This is mainly motivated by many real world

Table D.2: Estimation of number of parameters of Cai-2019. MSD: Multi-scale decomposition layer [174], IMSD: inverse MSD, Conv: 2D convolution layer, GDN: Generalized divisive normalization layer [175], IGDN: inverse GDN, Dconv: 2D deconvolution layer, ResNet: basic ResNet [15] block, ASPP: Atrous spatial pyramid pooling [21]

Module	Layer	Parameters	Total
Encoder	Conv-1	24,320	
	GDN	102,720	
	Conv-2	2,560,320	
	GDN	102,720	
	Conv-3	2560,320	
	GDN	102720	5,453,120
MSD	B1, Conv-1	921,920	
	B1, Conv-2	921,920	
	B1, Conv-3	921,920	
	B1, Conv-4	553,152	
	B1, GDN	37,056	
	B2, Conv-5	345,720	
	B2, GDN	345720	4,047,408
Implicit ROI prediction network	Conv-1	102,720	
	Conv-2	51,841	
	ResNet block		
	Conv-1	921,920	
	Conv-2	921,920	
	ASPP(1,3,6,9)		
	Conv-1	102,720	
	Conv-2	921,920	
	Conv-3	921,920	
	Conv-4	921,920	4,866,881
IMSD	B1, IGDN	37,056	
	B1, Dconv-1	553,280	
	B1, Dconv-2	921,920	
	B1, Dconv-3	921,920	
	B1, Dconv-4	921,920	
	B2, IGDN	1,4520	
	B2, Dconv-5	345920	3,716,536
Decoder	IGDN	102,720	
	Dconv-1	2,560,320	
	IGDN	102,720	
	Dconv-2	2,560,320	
	IGDN	102,720	
	Dconv-3	24,003	5,452,803
Cai-2019			23,536,748

applications where existing classifiers may either be shared with other modules or are costly to retrain. Nevertheless, our proposed framework is fully compatible for deployment and applications when fine-tuning of image classifier is possible.

In view of Sec. 6.4.1, we can minimize the same loss function in Eq. (6.10), but also activate classifier optimization along with other parts of the codec during training. In our test, we initialize the off-the-shelf classifier model from its pre-trained state and apply Adam optimizer for 40 training epochs with initial learning rate of 1e-4, reduced to $\times 1/10$, respectively, at 15 and 30 epochs.



(a) ROI PSNR (b) Overall MS-SSIM (c) Classification accuracy

Figure D.2: Rate-distortion and classification accuracy comparison with and without fine tuning the classifier for Imagenette dataset. Fine-tuning the classifier improves ROI PSNR and classification accuracy for all considered cases.

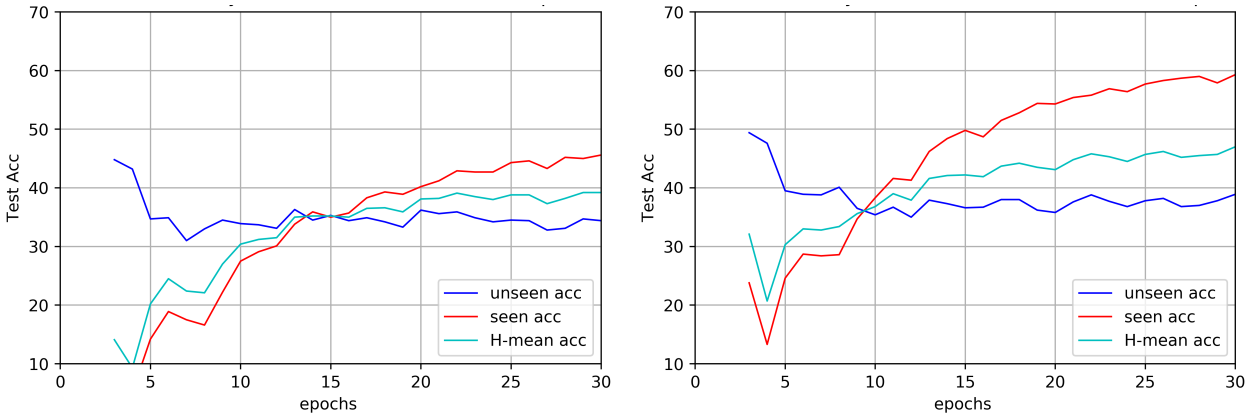
The resulting $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ for Imagenette is shown in Fig D.2. Clearly, classifier fine tuning can significantly improve classification accuracy for each scenario under consideration. Our proposed joint ROI compression and classification framework maintains similar benefits shown in Fig. 6.7. When feasible, fine-tuning the classifier further improves $\mathcal{R}$-$\mathcal{D}$-$\mathcal{C}$ performance for the proposed framework.

E

LDR-GUIDED LOW-SHOT LEARNING

E.1 CLASSES VERSUS ATTRIBUTES: AN ABLATION STUDY

In this section, we validate the importance of using attribute-based membership over class-based membership for ZSL when calculating MCR^2 loss as described in Sec. 7.3.1. For this purpose, we trained two LDR-guided ALE classifiers: (a) with $\mathbf{\Pi}$ which encodes seen class labels and (b) with $\mathbf{\Pi}$ which encodes seen attributes. To further isolate the membership function's ability to learn generalized features, we did not use ImageNet-1k pre-trained parameters to initialize the encoder. We then followed the same training steps as in Sec. 7.4.1. Fig. E.1 shows how the test accuracy of seen, unseen and H-mean categories change with epochs for AwA2 data set.



(a) Class-based membership (b) Attribute-based membership

Figure E.1: ZSL and GZSL accuracy comparison.

Note that compared to class-based membership, attribute-based membership records improved accuracy for all three accuracy categories. In particular, the unseen accuracy for attribute-based membership case continuous to improve even after 30 epochs whereas the useen accuracy of class-based membership case starts to drop after 20 epochs. This

observation indicates that over-fitting to seen classes is further mitigated with attribute-based membership for calculating MCR^2 loss during training of ZSL tasks.

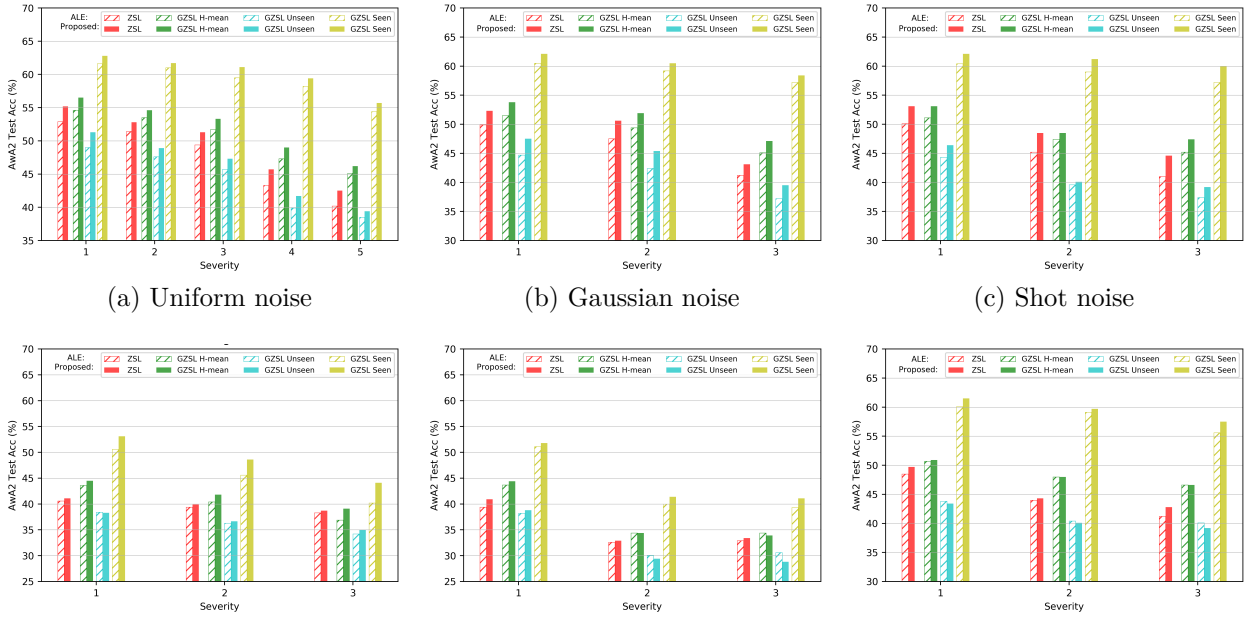
E.2 ROBUSTNESS TO NOISE AND COMMON PERTURBATIONS

We evaluate the robustness of the proposed LDR-guided ALE classifiers for different levels of uniform, Gaussian, and shot noises as well as fog, snow and defocus blur perturbations. Fig. E.3 shows visual illustrations of the sample images corrupted with noise and common perturbations. As revealed in the top row of Fig. E.2, our LDR-guided classifier demonstrates ZSL accuracy gain of at least 2% over the baseline model for each level and each type of additive noise. Besides, as demonstrated in the bottom row of Fig. E.2, for each of the considered perturbation cases, LDR-guided classifier exhibits marginal ZSL accuracy improvement over the baseline and almost zero H-mean accuracy gain under defocus, pixelation and snow perturbations. Under fog perturbations, however, the accuracy improvement is evident under various severity levels.

E.3 ROBUSTNESS TO NUMBER OF UNSEEN CLASSES

Fig. E.4 compares 5-way and 2-way ZSL accuracy of the proposed LDR-guided ALE classifier against baseline ALE classifier on corrupted images with uniform noise. When reporting accuracy for each 5-way or 2-way case, we averaged the test accuracy over all possible combination of test class selections. For instance, we averaged the test accuracy over ${}^{10}C_2$ test set combinations for 2-way classification for both baseline and the proposed method.

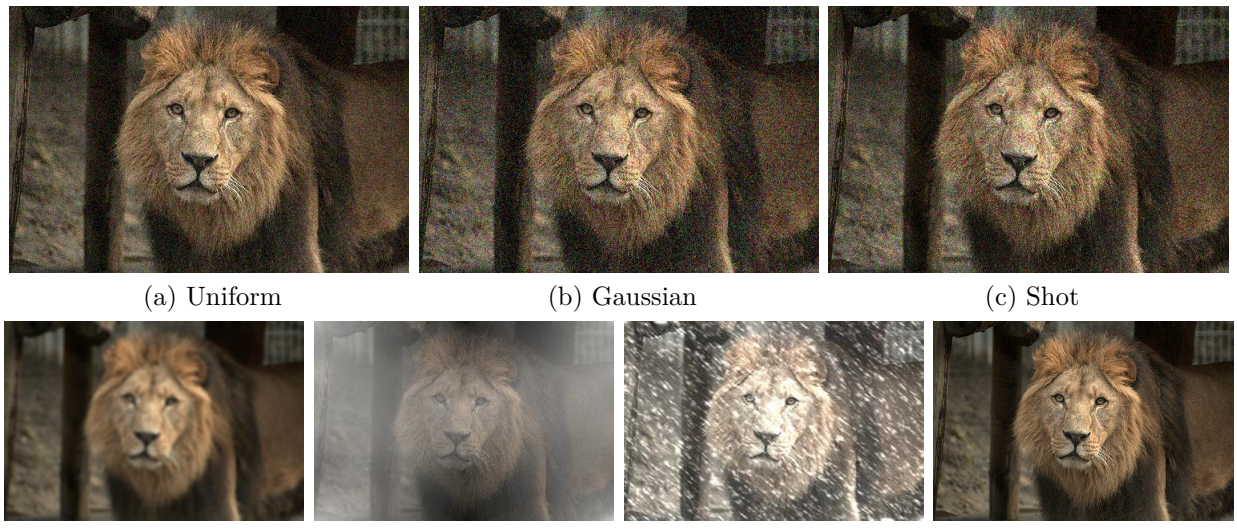
Note that significant ZSL accuracy gains can be observed for the proposed LDR-guided ALE over baseline ALE for both 2-way and 5-way cases, specially at lower severity levels of uniform noise.



(a) Uniform noise (b) Gaussian noise (c) Shot noise

(d) Fog (e) Snow (f) Defocus

Figure E.2: Noise severity vs accuracy performance of the proposed LDR-guided ALE model under (a) Uniform, (b) Gaussian and (c) Shot noises. And Perturbation severity vs accuracy performance of the proposed LDR-guided ALE model under (d) fog, (e) snow and (f) defocus blur.



(a) Uniform (b) Gaussian (c) Shot

(d) Defocus (e) Fog (f) Snow (g) Original

Figure E.3: Original image (g) is corrupted with noises: Uniform, Gaussian, and Shot noise, and common perturbations: defocus, fog and snow at severity level 2.

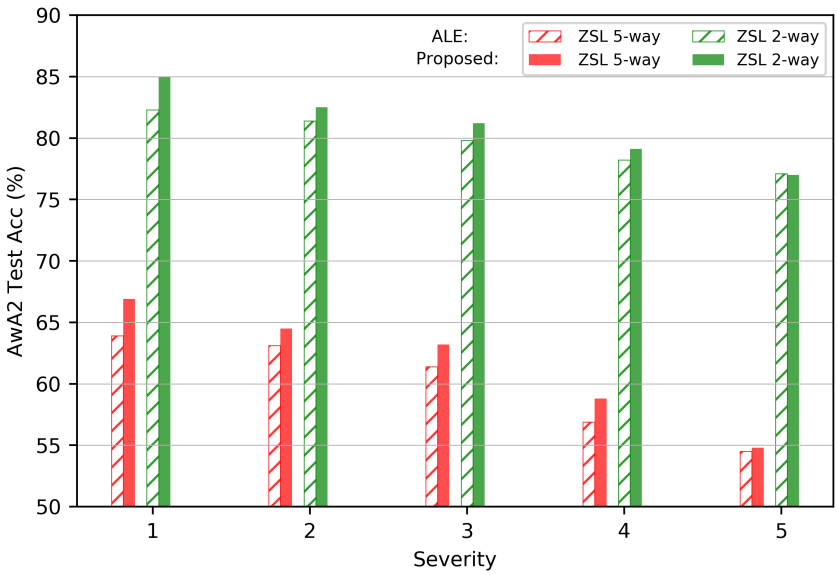


Figure E.4: ZSL accuracy comparison of the proposed LDR-guided ALE for 5-way and 2-way classification under uniform noise on AwA2 data set.

E.4 LDR-GUIDED FSL TRAINING

To ensure the reproducibility of the results reported by the proposed LDR-guided FSL classifiers, in Table. E.1, we list the values of learning rates (lr) and trade-off parameter λ used during training on Mini-ImageNet data set. Note that for a given encoder-shot combination, same $\{lr, \lambda\}$ values were used to produce the accuracy of LDR-guided versions of all FSL approaches reported in Table. 7.3.

Table E.1: LDR-guided FSL training details: learning rate and λ

	ConvNet		ResNet-12	
Parameter	1-shot	5-shot	1-shot	5-shot
Learning late	0.0001	0.0001	0.001	0.0002
λ	0.04	0.04	0.1	0.04

E.5 FSL ENCODER MODELS

In addition to popular encoder backbones used in recent FSL literature [138], we experimented with the following encoder backbones during ablation study in Sec. 7.4.2. Table. E.2 details model architectures of above encoders.

Table E.2: LDR-guided FSL encoder architectures

Conv-4		ResNet-10	
block	output	block	output
input	3×84×84	input	3×84×84
Conv (64, 3, 2)	64×42×42	Conv2d (64, 3, 1)	64×84×84
Conv (64, 3, 2)	64×21×21	ResNet (64, 3, 1)	64×84×84
Conv (64, 3, 2)	64×10×10	ResNet (128, 3, 2)	128×42×42
Conv (64, 3, 2)	64×5×5	ResNet (256, 3, 2)	256×21×21
		ResNet (512, 3, 2)	512×11×11
		AvgPool2d	512×1×1
Flatten	1600	Flatten	512

REFERENCES

[1] Y. Liu, J. Guo, D. Cai, and X. He, "Attribute attention for semantic disambiguation in zero-shot learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6698–6707.

[2] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[3] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE access*, vol. 7, pp. 53 040–53 065, 2019.

[4] S. Dörner, S. Cammerer, J. Hoydis, and S. Ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2017.

[5] H. Huang, S. Guo, G. Gui, Z. Yang, J. Zhang, H. Sari, and F. Adachi, "Deep learning for physical-layer 5g wireless techniques: Opportunities, challenges and solutions," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 214–222, 2019.

[6] S. Khan and T. Yairi, "A review on the application of deep learning in system health management," *Mechanical Systems and Signal Processing*, vol. 107, pp. 241–265, 2018.

[7] F. Piccialli, V. Di Somma, F. Giampaolo, S. Cuomo, and G. Fortino, "A survey on deep learning in medicine: Why, how and when?" *Information Fusion*, vol. 66, pp. 111–137, 2021.

[8] S. Nosratabadi, A. Mosavi, P. Duan, P. Ghamisi, F. Filip, S. S. Band, U. Reuter, J. Gama, and A. H. Gandomi, "Data science in economics: comprehensive review of advanced machine learning and deep learning methods," *Mathematics*, vol. 8, no. 10, p. 1799, 2020.

[9] Y. Zhao, J. Li, and L. Yu, "A deep learning ensemble approach for crude oil price forecasting," *Energy Economics*, vol. 66, pp. 9–16, 2017.

[10] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, "Evaluating explanation methods for deep learning in security," in *2020 IEEE european symposium on security and privacy (EuroS&P)*. IEEE, 2020, pp. 158–174.

[11] M. A. Amanullah, R. A. A. Habeeb, F. H. Nasaruddin, A. Gani, E. Ahmed, A. S. M. Nainar, N. M. Akim, and M. Imran, "Deep learning and big data technologies for iot security," *Computer Communications*, vol. 151, pp. 495–517, 2020.

[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[21] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[22] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[23] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[24] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, 2019.

[25] L. D. Chamain, S. S. Cheung, and Z. Ding, "Quannet: Joint image compression and classification over channels with limited bandwidth," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019, pp. 338–343.

[26] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, "IoT-based big data storage systems in cloud computing: perspectives and challenges," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 75–87, 2016.

[27] A. M. Ghosh and K. Grolinger, "Edge-cloud computing for internet of things data analytics: Embedding intelligence in the edge with deep learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2191–2200, 2020.

[28] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb 1992.

[29] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer Science & Business Media, 2012, vol. 642.

[30] L. D. Chamain and Z. Ding, "Improving deep learning classification of jpeg2000 images over bandlimited networks," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 4062–4066.

[31] L. D. Chamain, F. Racapé, J. Bégaint, A. Pushparaja, and S. Feltman, "End-to-end optimized image compression for machines, a study," in *2021 Data Compression Conference (DCC)*. IEEE, 2021, pp. 163–172.

[32] J. Chen, Y. Ye, and S. H. Kim, "JVET-Q2002 Algorithm description for Versatile Video Coding and Test Model 8 (VTM 8)," Jan. 2020.

[33] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," *arXiv preprint arXiv:1802.01436*, 2018.

[34] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Conditional probability models for deep image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4394–4402.

[35] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013.

[36] Y. Duan, Y. Zhang, X. Tao, C. Han, M. Xu, C. Yang, and J. Lu, "Content-aware deep perceptual image compression," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2019, pp. 1–6.

[37] H. Akutsu and T. Naruko, "End-to-end deep roi image compression," *IEICE Transactions on Information and Systems*, vol. 103, no. 5, pp. 1031–1038, 2020.

[38] C. Cai, L. Chen, X. Zhang, and Z. Gao, "End-to-end optimized roi image compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 3442–3457, 2019.

[39] C. De Alwis, A. Kalla, Q.-V. Pham, P. Kumar, K. Dev, W.-J. Hwang, and M. Liyanage, "Survey on 6g frontiers: Trends, applications, requirements, technologies and future research," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 836–886, 2021.

[40] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," *White Paper*, vol. 1, 2018.

[41] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE access*, vol. 6, pp. 6900–6919, 2017.

[42] F. Samie, L. Bauer, and J. Henkel, "From cloud down to things: An overview of machine learning in internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4921–4934, 2019.

[43] A. A. Diro, N. Chilamkurti, and Y. Nam, "Analysis of lightweight encryption scheme for fog-to-things communication," *IEEE Access*, vol. 6, pp. 26 820–26 830, 2018.

[44] C. Gong, F. Lin, X. Gong, and Y. Lu, "Intelligent cooperative edge computing in internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9372–9382, 2020.

[45] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[46] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[47] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C.-T. Lin, "Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment," *IEEE access*, vol. 6, pp. 1706–1717, 2017.

[48] G. Algan and I. Ulusoy, "Label noise types and their effects on deep learning," *arXiv preprint arXiv:2003.10471*, 2020.

[49] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *International conference on machine learning*. PMLR, 2018, pp. 4334–4343.

[50] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 3, pp. 453–465, 2013.

[51] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks." in *AAAI*, vol. 1, no. 2, 2008, p. 3.

[52] X. Dai, S. Tong, M. Li, Z. Wu, M. Psenka, K. H. R. Chan, P. Zhai, Y. Yu, X. Yuan, H.-Y. Shum *et al.*, "CTRL: Closed-loop transcription to an LDR via minimaxing rate reduction," *Entropy*, vol. 24, no. 4, p. 456, 2022.

[53] Y. Yu, K. H. R. Chan, C. You, C. Song, and Y. Ma, "Learning diverse and discriminative representations via the principle of maximal coding rate reduction," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[54] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2251–2265, 2018.

[55] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, "Latent embeddings for zero-shot classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 69–77.

[56] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[57] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the annual meeting of the cognitive science society*, vol. 33, no. 33, 2011.

[58] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016.

[59] L. D. Chamain, S. Qi, and Z. Ding, "An end-to-end learning architecture for efficient image encoding and deep learning," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 691–695.

[60] R. C. Gonzalez, R. E. Woods *et al.*, "Digital image processing [M]," *Publishing house of electronics industry*, vol. 141, no. 7, 2002.

[61] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," *SIAM journal on mathematical analysis*, vol. 29, no. 2, pp. 511–546, 1998.

[62] D. Taubman, "High performance scalable image compression with ebcot," *IEEE Transactions on image processing*, vol. 9, no. 7, pp. 1158–1170, 2000.

[63] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[64] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[65] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 215–10 224.

[66] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[67] A. Habibian, T. v. Rozendaal, J. M. Tomczak, and T. S. Cohen, "Video compression with rate-distortion autoencoders," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7033–7042.

[68] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[69] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy data coding and compression," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 9, pp. 1546–1562, 2007.

[70] K. H. R. Chan, Y. Yu, C. You, H. Qi, J. Wright, and Y. Ma, "Redunet: A white-box deep network from the principle of maximizing rate reduction," *Journal of Machine Learning Research*, 2022.

[71] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu, and J. Yosinski, "Faster neural networks straight from JPEG," in *Advances in Neural Information Processing Systems*, 2018, pp. 3937–3948.

[72] X. Zou, X. Xu, C. Qing, and X. Xing, "High speed deep networks based on discrete cosine transformation," in *IEEE Int. Conf. on Image Processing (ICIP)*, Oct 2014, pp. 5921–5925.

[73] T. Williams and R. Li, "Advanced image classification using wavelets and convolutional neural networks," in *15th IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*, Dec 2016, pp. 233–239.

[74] S. Fujieda, K. Takayama, and T. Hachisuka, "Wavelet convolutional neural networks for texture classification," *arXiv preprint arXiv:1707.07394*, 2017.

[75] A. Levinskis, "Convolutional neural network feature reduction using wavelet transform," *Elektronika ir Elektrotechnika*, vol. 19, no. 3, pp. 61–64, 2013.

[76] E. Kang, J. Min, and J. Ye, "A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction," *Medical Physics*, vol. 44, no. 10, 2017.

[77] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[78] R. Torfason, F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Towards image understanding from deep compression without decoding," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[79] J. Chao, H. Chen, and E. Steinbach, "On the design of a novel jpeg quantization table for improved feature detection performance," in *2013 IEEE Inte. Conf. on Image Processing*, Sept 2013, pp. 1675–1679.

[80] Z. Liu, T. Liu, W. Wen, L. Jiang, J. Xu, Y. Wang, and G. Quan, "DeepN-JPEG: A deep neural network favorable jpeg-based image compression framework," in *Proc. 55th IEEE Annual Design Automation Conf. (DAC'18*, San Francisco, USA, 2018, pp. 18:1–18:6. [Online]. Available: http://doi.acm.org/10.1145/3195970.3196022

[81] G. Chen, Y. Li, and S. N. Srihari, "Joint visual denoising and classification using deep learning," in *IEEE International Conference on Image Processing*, Sept 2016, pp. 3673–3677.

[82] S. Diamond, V. Sitzmann, F. Julca-Aguilar, S. Boyd, G. Wetzstein, and F. Heide, "Dirty pixels: Towards end-to-end image processing and perception," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 3, pp. 1–15, 2021.

[83] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[84] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[85] L. D. Chamain and Z. Ding, "Faster and accurate classification for jpeg2000 compressed images in networked applications," *arXiv preprint arXiv:1909.05638*, 2019.

[86] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking, "A review on internet of things (iot), internet of everything (ioe) and internet of nano things (iont)," in *2015 Internet Technologies and Applications (ITA)*. IEEE, 2015, pp. 219–224.

[87] Q. Rao and J. Frtunikj, "Deep learning for self-driving cars: chances and challenges," in *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, 2018, pp. 35–38.

[88] G. Sreenu and M. S. Durai, "Intelligent video surveillance: a review through deep learning techniques for crowd analysis," *Journal of Big Data*, vol. 6, no. 1, p. 48, 2019.

[89] Z. Yang and L. S. Pun-Cheng, "Vehicle detection in intelligent transportation systems and its applications under varying environments: A review," *Image and Vision Computing*, vol. 69, pp. 143–154, 2018.

[90] J. Kaur, M. A. Khan, M. Iftikhar, M. Imran, and Q. E. U. Haq, "Machine learning techniques for 5g and beyond," *IEEE Access*, vol. 9, pp. 23 472–23 488, 2021.

[91] B. Ji, Y. Wang, K. Song, C. Li, H. Wen, V. G. Menon, and S. Mumtaz, "A survey of computational intelligence for 6g: Key technologies, applications and trends," *IEEE Transactions on Industrial Informatics*, 2021.

[92] H. Wang, C. Tao, J. Qi, H. Li, and Y. Tang, "Semi-supervised variational generative adversarial networks for hyperspectral image classification," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2019, pp. 9792–9794.

[93] Y. Luo and H. Pfister, "Adversarial defense of image classification using a variational auto-encoder," *arXiv preprint arXiv:1812.02891*, 2018.

[94] X. Chen, Y. Sun, M. Zhang, and D. Peng, "Evolving deep convolutional variational autoencoders for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 815–829, 2020.

[95] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 eighth international conference on quality of multimedia experience (QoMEX)*. IEEE, 2016, pp. 1–6.

[96] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6316136/

[97] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[98] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," 2017.

[99] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00339

[100] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *International Conference on Learning Representations (ICLR)*, 2018.

[101] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5306–5314.

[102] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[103] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*. IEEE, 2015, pp. 1–5.

[104] S. Singh, S. Abu-El-Haija, N. Johnston, J. Ballé, A. Shrivastava, and G. Toderici, "End-to-end learning of compressible features," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 3349–3353.

[105] X. Luo, H. Talebi, F. Yang, M. Elad, and P. Milanfar, "The rate-distortion-accuracy tradeoff: Jpeg case study," *arXiv preprint arXiv:2008.00605*, 2020.

[106] M. Weber, C. Renggli, H. Grabner, and C. Zhang, "Lossy image compression with recurrent neural networks: from human perceived visual quality to classification accuracy," *arXiv preprint arXiv:1910.03472*, 2019.

[107] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrell, and Z. Akata, "Generalized zero-and few-shot learning via aligned variational autoencoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8247–8255.

[108] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the h. 264/avc video compression standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 620–636, 2003.

[109] V. Sze and D. Marpe, "Entropy coding in hevc," in *High Efficiency Video Coding (HEVC)*. Springer, 2014, pp. 209–274.

[110] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Practical full resolution learned lossless image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 629–10 638.

[111] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, 2014.

[112] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[113] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[114] Y. Deng, "Deep learning on mobile devices: a review," in *Mobile Multimedia/Image Processing, Security, and Applications 2019*, vol. 10993. International Society for Optics and Photonics, 2019, p. 109930A.

[115] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar, "Squeezing deep learning into mobile and embedded devices," *IEEE Pervasive Computing*, vol. 16, no. 3, pp. 82–88, 2017.

[116] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[117] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, and J. Jiang, "A simple pooling-based design for real-time salient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[118] Z. Chen, Q. Xu, R. Cong, and Q. Huang, "Global context-aware progressive aggregation network for salient object detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 10 599–10 606.

[119] Y. Ma, Y. Zhai, C. Yang, J. Yang, R. Wang, J. Zhou, K. Li, Y. Chen, and R. Wang, "Variable rate roi image compression optimized for visual quality," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1936–1940.

[120] P. Sreenivasulu and S. Varadarajan, "An efficient lossless roi image compression using wavelet-based modified region growing algorithm," *Journal of Intelligent Systems*, vol. 1, no. ahead-of-print, 2018.

[121] A. M. Taheri and H. Mahdavi-Nasab, "Sparse representation based facial image compression via multiple dictionaries and separated roi," *Multimedia Tools and Applications*, vol. 77, no. 23, pp. 31 095–31 114, 2018.

[122] W. Wang, Q. Lai, H. Fu, J. Shen, H. Ling, and R. Yang, "Salient object detection in the deep learning era: An in-depth survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[123] J. Choi, H. J. Chang, T. Fischer, S. Yun, K. Lee, J. Jeong, Y. Demiris, and J. Y. Choi, "Context-aware deep feature compression for high-speed visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 479–488.

[124] B. Wang, Q. Peng, E. Wang, K. Han, and W. Xiang, "Region-of-interest compression and view synthesis for light field video streaming," *IEEE Access*, vol. 7, pp. 41 183–41 192, 2019.

[125] A. I. Khaleel, N. A. H. Zahri, and M. I. Ahmad, "A hybrid compression method for medical images based on region of interest using artificial neural networks," *Journal of Engineering*, vol. 2021, 2021.

[126] E. M. Rubino, A. J. Álvares, R. Marín, and P. J. Sanz, "Real-time rate distortion-optimized image compression with region of interest on the arm architecture for underwater robotics applications," *Journal of Real-Time Image Processing*, vol. 16, no. 1, pp. 193–225, 2019.

[127] G. Li and Y. Yu, "Deep contrast learning for salient object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 478–487.

[128] N. Liu and J. Han, "Dhsnet: Deep hierarchical saliency network for salient object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 678–686.

[129] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr, "Deeply supervised salient object detection with short connections," *IEEE TPAMI*, vol. 41, no. 4, pp. 815–828, 2019.

[130] Benjamin Bross, Jianle Chen, Shan Liu, and Y.-K. Wang, "JVET-S2001 Versatile Video Coding (Draft 10)," in *Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11*, Oct. 2020.

[131] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, "Compressai: a pytorch library and evaluation platform for end-to-end compression research," *arXiv preprint arXiv:2011.03029*, 2020.

[132] S. Liu, M. Long, J. Wang, and M. I. Jordan, "Generalized zero-shot learning with deep calibration network," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[133] M. Rohrbach, M. Stark, and B. Schiele, "Evaluating knowledge transfer and zero-shot learning in a large-scale setting," in *CVPR 2011*. IEEE, 2011, pp. 1641–1648.

[134] X. Xu, F. Shen, Y. Yang, D. Zhang, H. Tao Shen, and J. Song, "Matrix tri-factorization with manifold regularizations for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3798–3807.

[135] L. Zhang, T. Xiang, and S. Gong, "Learning a deep embedding model for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2021–2030.

[136] W. Xu, Y. Xian, J. Wang, B. Schiele, and Z. Akata, "Attribute prototype network for zero-shot learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 969–21 980, 2020.

[137] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[138] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, "Few-shot learning via embedding adaptation with set-to-set functions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8808–8817.

[139] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha, "An empirical study and analysis of generalized zero-shot learning for object recognition in the wild," in *European conference on computer vision*. Springer, 2016, pp. 52–68.

[140] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata, "Feature generating networks for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5542–5551.

[141] Y. Zhu, M. Elhoseiny, B. Liu, X. Peng, and A. Elgammal, "A generative adversarial approach for zero-shot learning from noisy texts," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1004–1013.

[142] Y. Zhu, J. Xie, B. Liu, and A. Elgammal, "Learning feature-to-feature translator by alternating back-propagation for generative zero-shot learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9844–9854.

[143] S. Tong, X. Dai, Z. Wu, M. Li, B. Yi, and Y. Ma, "Incremental learning of structured memory via closed-loop transcription," *arXiv preprint arXiv:2202.05411*, 2022.

[144] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for image classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 7, pp. 1425–1438, 2015.

[145] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[146] H. Li, D. Eigen, S. Dodge, M. Zeiler, and X. Wang, "Finding task-relevant features for few-shot learning by category traversal," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1–10.

[147] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 951–958.

[148] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[149] G. Patterson, C. Xu, H. Su, and J. Hays, "The sun attribute database: Beyond categories for deeper scene understanding," *International Journal of Computer Vision*, vol. 108, no. 1, pp. 59–81, 2014.

[150] G.-S. Xie, L. Liu, X. Jin, F. Zhu, Z. Zhang, J. Qin, Y. Yao, and L. Shao, "Attentive region embedding network for zero-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9384–9393.

[151] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*.

[152] L. D. Chamain, S.-c. S. Cheung, and Z. Ding, "Quannet: Joint image compression and classification over channels with limited bandwidth," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 338–343.

[153] L. D. Chamain and Z. Ding, "Improving deep learning classification of jpeg2000 images over bandlimited networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4062–4066.

[154] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, "Learning with noisy labels." in *NIPS*, vol. 26, 2013, pp. 1196–1204.

[155] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5552–5560.

[156] H. Wei, Y. Yang, L. Li, A. B. Winston, and A. Ten-Ami, "Hybrid learning for adaptive video grouping and compression," Sep. 17 2019, uS Patent 10,419,773.

[157] K.-O. Cheng, N.-F. Law, and W.-C. Siu, "Clustering-based compression for population dna sequences," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 16, no. 1, pp. 208–221, 2017.

[158] R. Kozhemiakin, S. Abramov, V. Lukin, B. Djurović, I. Djurović, and B. Vozel, "Lossy compression of landsat multispectral images," in *2016 5th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2016, pp. 104–107.

[159] X. Zhu and M. Bain, "B-cnn: branch convolutional neural network for hierarchical classification," *arXiv preprint arXiv:1709.09890*, 2017.

[160] J. Y. Chang and K. M. Lee, "Large margin learning of hierarchical semantic similarity for image classification," *Computer Vision and Image Understanding*, vol. 132, pp. 3–11, 2015.

[161] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[162] L. D. Chamain, F. Racapé, J. Bégaint, A. Pushparaja, and S. Feltman, "End-to-end optimized image compression for multiple machine tasks," *arXiv preprint arXiv:2103.04178*, 2021.

[163] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[164] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[165] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5455–5463.

[166] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 724–732.

[167] A. Gomez Chavez, A. Ranieri, D. Chiarella, E. Zereik, A. Babić, and A. Birk, "Caddy underwater stereo-vision dataset for human–robot interaction (hri) in the context of diver activities," *Journal of Marine Science and Engineering*, vol. 7, no. 1, p. 16, 2019.

[168] A. G. Chavez, A. Ranieri, D. Chiarella, and A. Birk, "Underwater vision-based gesture recognition: A robustness validation for safe human-robot interaction," *IEEE Robotics & Automation Magazine*, 2021.

[169] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[170] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.

[171] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[172] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[173] H. W. Kuhn, "Nonlinear programming: a historical view," in *Traces and Emergence of Nonlinear Programming*. Springer, 2014, pp. 393–414.

[174] C. Cai, L. Chen, X. Zhang, and Z. Gao, "Efficient variable rate image compression with multi-scale decomposition network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 12, pp. 3687–3700, 2018.

[175] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.